Volume 23 Number 3

# ACTA CYBERNETICA



Szeged, 2018

# ACTA CYBERNETICA

**Information for authors.** Acta Cybernetica publishes only original papers in the field of Computer Science. Manuscripts must be written in good English. Contributions are accepted for review with the understanding that the same work has not been published elsewhere. Papers previously published in conference proceedings, digests, preprints are eligible for consideration provided that the author informs the Editor at the time of submission and that the papers have undergone substantial revision. If authors have used their own previously published material as a basis for a new submission, they are required to cite the previous work(s) and very clearly indicate how the new submission offers substantively novel or different contributions beyond those of the previously published work(s). There are no page charges. An electronic version of the published paper is provided for the authors in PDF format.

**Manuscript Formatting Requirements.** All submissions must include a title page with the following elements: title of the paper; author name(s) and affiliation; name, address and email of the corresponding author; an abstract clearly stating the nature and significance of the paper. Abstracts must not include mathematical expressions or bibliographic references.

References should appear in a separate bibliography at the end of the paper, with items in alphabetical order referred to by numerals in square brackets. Please prepare your submission as one single PostScript or PDF file including all elements of the manuscript (title page, main text, illustrations, bibliography, etc.).

When your paper is accepted for publication, you will be asked to upload the complete electronic version of your manuscript. For technical reasons we can only accept files in LaTeX format. It is advisable to prepare the manuscript following the guidelines described in the author kit available at `http://www.inf.u-szeged.hu/kutatas/acta-cybernetica/information-for-authors#AuthorKit` even at an early stage.

**Submission and Review.** Manuscripts must be submitted online using the editorial management system at `http://cyber.bibl.u-szeged.hu/index.php/actcybern/submission/wizard`. Each submission is peer-reviewed by at least two referees. The length of the review process depends on many factors such as the availability of an Editor and the time it takes to locate qualified reviewers. Usually, a review process takes 6 months to be completed.

**Subscription Information.** Acta Cybernetica is published by the Institute of Informatics, University of Szeged, Hungary. Each volume consists of four issues, two issues are published in a calendar year. Subscription rates for one issue are as follows: 5000 Ft within Hungary, €40 outside Hungary. Special rates for distributors and bulk orders are available upon request from the publisher. Printed issues are delivered by surface mail in Europe, and by air mail to overseas countries. Claims for missing issues are accepted within six months from the publication date. Please address all requests to:

Acta Cybernetica, Institute of Informatics, University of Szeged
P.O. Box 652, H-6701 Szeged, Hungary
Tel: +36 62 546 396, Fax: +36 62 546 397, Email: `acta@inf.u-szeged.hu`

**Web access.** The above information along with the contents of past and current issues are available at the Acta Cybernetica homepage `https://www.inf.u-szeged.hu/en/kutatas/acta-cybernetica` .

**László Lovász**
Department of Computer Science
Eötvös Loránd University
Budapest, Hungary
lovasz@cs.elte.hu

**Dana Petcu**
Department of Computer Science
West University of Timisoara, Romania
petcu@info.uvt.ro

**Heiko Vogler**
Department of Computer Science
Dresden University of Technology
Dresden, Germany
Heiko.Vogler@tu-dresden.de

**Gerhard J. Woeginger**
Department of Mathematics and
Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
gwoegi@win.tue.nl

**Csanád Imreh (1975–2017)**

# In memoriam Csanád Imreh (1975-2017)

The present volume is a collection of papers dedicated to the memory of Professor Csanád Imreh, who passed away unexpectedly in 2017. These papers were written by his friends, colleagues and former students.

Csanád Imreh was born in Szeged on 20th May in 1975. His interest in mathematics emerged quite early. He started to work for the "Középiskolai Matematikai Lapok" "Mathematical letters for secondary schools", a special journal for mathematically talented secondary school students) at the upper elementary school. He continued to participate in different competitions when he was a secondary school student. He attended a special class for mathematically talented youngsters in Szeged.

At the university he studied mathematics, but he was already interested in algorithmic problems. At our faculty he was once elected for the most excellent student in mathematics. He also received a Pro Scientia award that is given for the best work at a nationwide competition for university students. He was then admitted to the Computer Science PhD program where he was among the very few students who finished his thesis within the regular three years of the scholarship in 2001. He received the habilitation title in 2010. Before 2010 he was assistant at the department of computer algorithms and artificial intelligence, then in 2010 he became Professor and the Head of the department.

His research interests were very broad: he had results among others in online algorithms, in combinatorial optimization, in scheduling and in automata theory. He published 56 papers in total. He applied for the title "Doctor of Sciences" at the Hungarian Academy of Sciences in 2015, but this process could never be completed. Professor Imreh had wide international academic connections. He was a TEMPUS scholar in Utrecht, then he spent 6 months as a visiting scholar in Graz. In 2001-2002 he spent a year as a postdoc fellow at the Max Planck Institute in Saarbrücken. In 2007-2008, he was a visiting scholar at the University of Kyoto. In 2012 and 2015 he was a Humboldt fellow in Berlin. He participated in organizing international conferences and served as a referee for well-known journals.

He was an active participant in cooperation with Hungarian researchers and in different organizations. He received the Pro Scientia prize, the Kalmár László prize and the Farkas Gyula prize. He was also awarded a Békési György scholarship and a Bolyai János scholarship. He participated in different projects financed by OTKA, NKFP and TÁMOP. He was secretary of the Computer Science Doctoral School of our university, and member of the Informatics and Computer Science Committee of the Hungarian Academy of Sciences. He was also editor of Acta Cybernetica, an international journal published by the Institute of Computer Science at the University of Szeged.

Professor Imreh was an outstanding, internationally acknowledged personality of our Institute. He was hard working and at the same time good-humored and even-handed. Without doubt, he had great potential and a great future ahead of him.

We miss you, Csanád.

Szeged, June 2018

János Csirik
Department of Computer Science
University of Szeged, Hungary

Acta Cybernetica **23** (2018) 761–772.

# The Metric Dimension of Two-Dimensional Extended Meshes

Ron Adar[a] and Leah Epstein[b]

**Abstract**

We consider two-dimensional grids with diagonals, also called extended meshes or meshes. Such a graph consists of vertices of the form $(i, j)$ for $1 \leq i \leq m$ and $1 \leq j \leq n$, for given $m, n \geq 2$. Two vertices are defined to be adjacent if the $\ell_\infty$ distance between their vectors is equal to 1. A landmark set is a subset of vertices $L \subseteq V$, such that for any distinct pair of vertices $u, v \in V$, there exists a vertex of $L$ with different distances to $u$ and $v$. We analyze the metric dimension and show how to obtain a landmark set of minimum cardinality.

**Keywords:** metric dimension, mesh graph, grid graph, landmark set, resolving set

## 1 Introduction

Consider an undirected graph $G = (V, E)$. For $u, v \in V$, let $d(u, v)$ denote the edge distance between these two vertices. A vertex $x \in V$ separates $u$ and $v$ if $d(x, u) \neq d(x, v)$, and in this case, $x$ is also called a separating vertex for $u$ and $v$. A landmark set (LS) is a subset $L \subseteq V$ such that for any pair of vertices $u \neq v$, $L$ has at least one vertex $y$ that separates $u$ and $v$. The vertices of a landmark set $L$ are often referred to as *landmarks*. In the algorithmic metric dimension problem, the goal is to find a landmark set $L$ of minimum cardinality. The cardinality of a minimum cardinality landmark set of $G$ is called the *metric dimension* of $G$. In a variant of the metric dimension problem (called the weighted metric dimension problem) the goal is to find a landmark set $L$ of minimum cost, where there is a non-negative cost function on $G$'s vertices.

[a]Department of Computer Science, University of Haifa, Haifa, Israel. E-mail: radar03@csweb.haifa.ac.il

[b]Department of Mathematics, University of Haifa, Haifa, Israel. E-mail: lea@math.haifa.ac.il

DOI: 10.14232/actacyb.23.3.2018.2

A two-dimensional extended mesh (or just mesh) $M_{m,n}$ (for integer parameters $m$ and $n$) has $|V| = m \cdot n$ vertices of the form $(i, j)$, where $1 \leq i \leq m$ and $1 \leq j \leq n$. For vertices $(i_1, j_1)$, $(i_2, j_2)$, let $((i_1, j_1), (i_2, j_2)) \in E$ if (and only if) $\max\{|i_1 - i_2|, |j_1 - j_2|\} = 1$. The resulting distance between two vertices is the $\ell_\infty$ distance between their vectors, that is, $d((i_1, j_1), (i_2, j_2)) = \max\{|i_1 - i_2|, |j_1 - j_2|\}$. This distance also called the chessboard distance. In accordance with the way we have defined the edges of the graph, it is the edge distance of $M_{m,n}$.

We number the rows of the mesh from top to bottom, and the columns from left to right. The rows are called $R_1$, $R_2$, ..., $R_m$ ($R_i$ is also called row $i$), and the columns are called $C_1$, $C_2$, ..., $C_m$ ($C_j$ is also called column $j$). The $j$th vertex of row $i$ is denoted by $(i, j)$.

A graph related to the two-dimensional mesh is the two-dimensional lattice, $G_{m,n}$, where there is an edge between vertices $(i_1, j_1)$, $(i_2, j_2)$, if (and only if) $|i_1 - i_2| + |j_1 - j_2| = 1$, and the resulting distance between two vertices is the $\ell_1$ distance between their vectors, that is, $d((i_1, j_1), (i_2, j_2)) = |i_1 - i_2| + |j_1 - j_2|$. This distance also called the city block distance.

Former results involving mesh graph and lattice graph (both are also called grid graphs) can be found in [15, 14, 16, 13, 2, 1].

We survey some known results on mesh graphs. In the case $m = 1$ and $n > 1$ (or $n = 1$ and $m > 1$), $G_{m,n}$ is a path, and its metric dimension is known to be 1 [14]. In the case $m = n = 1$, the graph has a single vertex, so the metric dimension is zero by definition. In [15], it was shown that for a two-dimensional lattice the metric dimension is always 2 (for $n \geq m \geq 2$), and for a two-dimensional mesh and $m = n \geq 2$, the metric dimension is 3. In our previous work on two-dimensional lattice with a cost function on its vertices [1], we described a polynomial time algorithm for solving the weighted metric dimension problem.

In this work, we will extend the result in [15], and calculate the metric dimension of a two-dimensional mesh, for all values of $m, n \geq 2, m \neq n$. The metric dimension of $M_{m,n}$, denoted by $MD(M_{m,n})$ is the minimum cardinality of any LS for $M_{m,n}$.

In this work our main goal is to prove the following theorem.

**Theorem 1.** *The metric dimension of a mesh $M_{m,n}$ with $n > m \geq 2$ satisfies $MD(M_{m,n}) = \lceil \frac{n-1}{m-1} \rceil + 1$.*

We will prove a tight bound of $\lceil \frac{n-1}{m-1} \rceil + 1$ on $MD(M_{m,n})$. The lower bound is proved via a direct analysis of all possible column landmark sets (see formal definition on the next section), and for the upper bound we describe a specific landmark set of $M_{m,n}$, whose cardinality is $\lceil \frac{n-1}{m-1} \rceil + 1$.

In the case where $n-1$ is divisible by $m-1$, the mesh contains $\frac{n-1}{m-1}$ concatenated square sub-meshes with $m$ rows and $m$ columns (where every two such sub-meshes

share a single column). In this case we will show that each square sub-mesh (with $m$ rows and $m$ columns) has exactly two landmark vertices in a minimum cardinality LS. Note that substituting $m = n$ does not give the correct metric dimension for $m = n$ (which is 3 as stated above, and not 2), and this is a special case. Informally, the reason for this property is that once $M_{m,n}$ is not a square (i.e., $m \neq n$), given two vertices contained in a square sub-mesh of $M_{m,n}$, they can be separated by a vertex outside this sub-mesh, in some cases.

Interestingly, as a result of Theorem 1, one of the differences between a two-dimensional lattice and a two-dimensional mesh, is that while the metric dimension of two-dimensional lattice is always 2, as quoted above, the metric dimension of two-dimensional mesh (for $n > m \geq 2$) grows as a function of $n$.

The first articles on the metric dimension problem were by Harary and Melter [11] and by Slater [18]. The problem is NP-hard [14] and hard to approximate [4, 9] for general graphs, and it was studied for specific graph classes [11, 18, 14, 6, 3, 17, 7, 5, 10]. Applications can be found in [4, 11, 15, 8, 14, 6], where some of these applications are relevant for weighted graphs (see also [10]).

## 2 The column metric dimension and a lower bound

In this section we define an auxiliary concept, which we use in order to provide a lower bound on $MD(M_{m,n})$.

A *Column Landmark Set (CLS)* is a set $L \subseteq V$ such that for any pair of distinct vertices on one column, $(i_1, j)$ and $(i_2, j)$ (where $i_1 \neq i_2$), there exists $u \in L$ such that $d(u, (i_1, j)) \neq d(u, (i_2, j))$. The *Column Metric Dimension* of a graph $M_{m,n}$ is the minimum cardinality of any CLS for it, and it is denoted by $CMD(M_{m,n})$.

Since any LS is a CLS (as all vertices are separated by some vertex of LS, this clearly holds for pairs of vertices on the same column), we have for any mesh $G$ that $MD(M_{m,n}) \geq CMD(M_{m,n})$. In this section we would like to prove a lower bound on $MD(M_{m,n})$, and for that we will prove the following lemma.

**Lemma 1.** *The column metric dimension of a mesh $M_{m,n}$ with $m, n \geq 2$ satisfies $CMD(M_{m,n}) \geq \lceil \frac{n-1}{m-1} \rceil + 1$. Thus, in the case $n > m$, we have $MD(M_{m,n}) \geq \lceil \frac{n-1}{m-1} \rceil + 1$.*

Note that the lower bound on $CMD(M_{m,n})$ is proved for any $m, n \geq 2$ and not only for the case $n > m$. In what follows we consider these more general cases. Before we prove the lemma, we state and prove several simple and useful claims. Consider specific values $m, n \geq 2$, and a specific CLS $L$ for this graph. Let $\mu_j$ be the number of elements of $L$ in column $C_j$ (that is $\mu_j = |C_j \cap L|$),

and let $N_j = \sum_{\ell=1}^{j} \mu_j$ be the number of landmarks in the first $j$ columns (i.e., $N_j = |(C_1 \cup C_2 \cup \cdots \cup C_j) \cap L|$). We let $N_j = N_n$ for $j > n$.

The goal of the first claim is to provide a lower bound on the number of landmarks in the first few columns. We will use this claim for the last $j$ columns as well (this follows from symmetry).

**Claim 1.** *Consider vertices $u_1 = (x_1, y')$ and $u_2 = (x_2, y')$ (on column $y'$). Let $v = (x, y)$ be a vertex that separates them, that is $d(v, u_1) \neq d(v, u_2)$. Then, $|y - y'| \leq m - 2$.*

*Proof.* Assume by contradiction that $|y-y'| \geq m-1$. Then, for $k = 1, 2$, $d(v, u_k) = \max\{|x - x_k|, |y - y'|\} = |y - y'|$, as $|x - x_k| \leq m - 1$ while $|y - y'| \geq m - 1$, so the distances to $v$ are determined by the columns. This contradicts $d(v, u_1) \neq d(v, u_2)$. □

**Claim 2.** *The inequality $N_{\lceil \frac{m}{2} \rceil} \geq 1$ holds for any CLS $L$. Moreover, if $N_1 = 0$, then $N_{m-1} \geq 2$.*

*Proof.* If $N_1 > 0$, the first part holds, so we assume $N_1 = 0$ and prove both parts of the claim. Consider two vertices on column 1, $a_1 = (\lfloor \frac{m}{2} \rfloor, 1)$ and $a_2 = (\lfloor \frac{m}{2} \rfloor + 1, 1)$ (for even values of $m$ they are exactly in the middle of column 1, for odd values of $m$ they are approximately in the middle of column 1). Note that $\lfloor \frac{m}{2} \rfloor \geq 1$ and $\lfloor \frac{m}{2} \rfloor + 1 \leq m$, for $m \geq 2$, so these two vertices are well defined.

Let $b = (x_b, y_b)$ denote a vertex of $L$ separating $a_1$ and $a_2$. We will show $y_b \leq \lceil \frac{m}{2} \rceil$, proving $N_{\lceil \frac{m}{2} \rceil} \geq 1$. Assume by contradiction $y_b \geq \lceil \frac{m}{2} \rceil + 1$. We will show that the distance of $b$ to $a_1$ and $a_2$ is defined by the difference between column indices and therefore the distances are equal. We get $d(a_1, b) = \max\{|x_b - \lfloor \frac{m}{2} \rfloor|, |y_b - 1|\}$ and $d(a_2, b) = \max\{|x_b - (\lfloor \frac{m}{2} \rfloor + 1)|, |y_b - 1|\}$. Since $1 \leq x_b \leq m$, we get $|x_b - \lfloor \frac{m}{2} \rfloor| \leq \lceil \frac{m}{2} \rceil \leq y_b - 1$ and $|x_b - \lfloor \frac{m}{2} \rfloor - 1| \leq \lceil \frac{m}{2} \rceil \leq y_b - 1$, so $d(a_1, b) = d(a_2, b) = y_b - 1$, a contradiction.

Next, we analyze $N_{m-1}$ for the case $N_1 = 0$. Since $N_{m-1} > 0$ (as $m - 1 \geq \lfloor \frac{m}{2} \rfloor$ for $m \geq 2$), there is at least one vertex $c \in L$ in this case, as we have shown. It remains to show that there is a pair of vertices in the first $m - 1$ columns not separated by $c$, implying $|L| \geq 2$. Let $c = (x_c, y_c)$ be a vertex of $L$ such that $2 \leq y_c \leq m - 1$. Once again we consider two vertices of the first column, and show that $c$ does not separate them. Let $a'_1 = (x_c, 1)$, i.e., the vertex of the first column on the same row as $c$. If $x_c = 1$, let $a'_2 = (x_c + 1, 1)$ and otherwise $a'_2 = (x_c - 1, 1)$. Thus, $a'_2$ is well defined, as it is either the vertex just above $a'_1$ or just below it (since $m \geq 2$, at least one of these vertices exists). We have $d(a'_1, c) = \max\{0, |y_c - 1|\} = y_c - 1$ and $d(a'_2, c) = \max\{1, |y_c - 1|\} = y_c - 1$, since $y_c \geq 2$. Thus, as $L$ contains a vertex separating $a'_1$ and $a'_2$, and by Claim 1 this

vertex is on one of the first $m-1$ columns (or of the $n$ columns, if $n < m-1$), we get $N_{m-1} \geq 2$. □

**Claim 3.** *Let $L$ be a CLS defined for $M_{m,n}$ such that $m, n \geq 2$ (where each element of $L$ is of the form $(i, j)$). If $L \cap C_n = \emptyset$, then $L$ is a CLS for $M_{m,n-1}$. If $L \cap R_m = \emptyset$, then $L$ is a CLS for $M_{m-1,n}$.*

*Proof.* Consider two vertices $(i_1, j_1)$ and $(i_2, j_2)$. Any shortest path between these two vertices traverses only vertices on columns $\min\{j_1, j_2\}$, ..., $\max\{j_1, j_2\}$ and rows $\min\{i_1, i_2\}$, ..., $\max\{i_1, i_2\}$, that is, on columns and rows between the columns of these vertices and rows between the rows of these vertices. This implies the validity of the claim. □

*Proof.* We now prove Lemma 1. We start this proof with several simple cases, which will allow us to use induction for the remaining cases (on $n + m$).

**Case 1.** Consider the case $m = 2$. In this case we show $CMD(M_{m,n}) \geq n$. To prove this, we show that $\mu_j \geq 1$ for $1 \leq j \leq n$. By Claim 1, the only vertices that separate the two vertices $(1, j)$ and $(2, j)$ are on column $j$, that is, one of these two vertices. Therefore, any CLS either contains at least one of $(1, j)$ and $(2, j)$.

**Case 2.** Consider the case $n \leq m$ (where $m \geq 3$). In this case we show $CMD(M_{m,n})) \geq 2$. If $\mu_1 \geq 1$ and $\mu_n \geq 1$, we are done. Otherwise, at least one of the columns $C_1$ and $C_n$ does not have a landmark. Assume without loss of generality (by rotating the mesh by 180 degrees or by reflecting it across a vertical line) that $L \cap C_1 = \emptyset$. By Claim 2, $N_{m-1} \geq 2$, so $|L| \geq 2$.

We are left with the case $n > m \geq 3$. We say that a *gap* (with respect to a CLS $L$) is a sequence of $m-1$ columns that do not contain elements of $L$, that is, there is an index $1 \leq k \leq n-m+2$ such that $\{C_k, C_{k+1}, \ldots, C_{k+m-2}\} \cap L = \emptyset$. The cases $k = 1$ and $k = n-m+2$ are not possible due to Claim 2, as the first $m-1$ columns have at least one element of $L$, and symmetrically, the last $m-1$ columns have at least one element of $L$. Thus, $k$ satisfies $2 \leq k \leq n-m+1$, and in particular, a gap is possible only if $n \geq m+1$ (since all cases where $n \leq m$ were already considered, a gap is defined for all remaining cases).

We use the first two cases as the induction base, and prove the remaining cases via induction.

**Case 3.** There is a gap in $L$. Let $k$ ($2 \leq k \leq n-m+1$) be such that $\{C_k, C_{k+1}, \ldots, C_{k+m-2}\} \cap L = \emptyset$, that is, all elements of $L$ are on columns $C_1, C_2 \ldots, C_{k-1}, C_{k+m-1}, \ldots, C_n$. Let $L_1 = \{C_1, C_2, \ldots, C_k\} \cap L$ and let $L_2 = \{C_{k+m-2}, C_{k+m-1}, \ldots, C_n\} \cap L$ (so $L = L_1 \cup L_2$ and $L_1 \cap L_2 = \emptyset$ since $m \geq 3$). We claim that $L_1$ is a CLS for the sub-mesh of $k$ columns and $m$ rows consisting of the first $k \geq 2$ columns. Since $L$ is a CLS, for every pair of distinct vertices $v_1$ and $v_2$ on one of the first $k$ columns, there is a vertex of $L$ separating them. By Claim 1, such a vertex is not on columns $k+m-1, \ldots, n$. As columns $k, k+1, \ldots, k+m-2$ have no elements of $L$, there is an element of $L$ separating $v_1$ and $v_2$ on one of the columns $1, 2, \ldots, k-1$, that is, it is an element of $L_1$. Thus, $L_1$ is a CLS for a mesh of $k$ columns and $m$ rows. Analogously, it is possible to prove that $L_2$ is a CLS for a mesh of $n-m+3-k \geq 2$ columns and $m$ rows. Using induction, and as the numbers of columns of $L_1$ and $L_2$ are $k$ and $n-m-k+3$, respectively (with $m$ rows), we find $|L| = |L_1| + |L_2| \geq \lceil \frac{k-1}{m-1} \rceil + 1 + \lceil \frac{(n-m+3-k)-1}{m-1} \rceil + 1 \geq 2 + \lceil \frac{n-m+1}{m-1} \rceil = \lceil \frac{n}{m-1} \rceil + 1 \geq \lceil \frac{n-1}{m-1} \rceil + 1$, which holds by properties of rounding up.

We are left with the case where there is no gap in $L$. For an integer $c \geq 0$, let $h_c = m + (m-1)c$ and $h'_c = m - 1 + (m-1)c = h_c - 1$. Note that by their definitions it holds that $m \leq h_c$ and $m - 1 \leq h'_c$ for any value of $c$. We claim that in this case, if $h_c \leq n$, we have $N_{h_c} \geq c + 2$, and additionally, if $h'_c \leq n$ and $N_1 = 0$ hold, $N_{h'_c} \geq c + 2$. This is proved by induction on $c$. For $c = 0$, if $N_1 = 0$, by Claim 2, $N_m \geq N_{m-1} \geq 2$. Otherwise, $N_1 \geq 1$, and since there are no gaps, at least one of the columns $C_2, C_3, \ldots, C_m$ has an element of $L$, so $N_m \geq 2$. The inductive step is proved in a similar manner. If $N_{h_{c-1}} \geq c + 1$ for some integer $c \geq 1$, as there is no gap, there is at least one element of $L$ on columns $h_{c-1}+1, h_{c-1}+2, \ldots, h_{c-1}+m-1$. Since $h_{c-1} = m + (m-1)(c-1) = 1 + (m-1)c$ we get that $h_{c-1} + m - 1 = h_c$, and as a result $N_{h_c} \geq N_{h_{c-1}} + 1 \geq c + 2$. If $N_{h'_{c-1}} \geq c + 1$ for some $c \geq 1$, as there is no gap, there is at least one element of $L$ on columns $h'_{c-1}+1, h'_{c-1}+2, \ldots, h'_{c-1}+m-1$. Since $h'_{c-1} = h_{c-1} - 1 = (m-1)c$ we get that $h'_{c-1} + m - 1 = h_{c-1} - 1 = h'_c$, and as a result $N_{h'_c} \geq N_{h'_{c-1}} + 1 \geq c + 2$.

**Case 4.** There is an element of $L$ on the first column or on the last column (or both). Without loss of generality (by possibly rotating the mesh by 180 degrees) we assume $L \cap C_n \neq \emptyset$, i.e., $\mu_n > 0$. Let $c = \lceil \frac{n-1}{m-1} \rceil - 2$, where $c \geq 0$ as $n > m$. Consider columns $1, 2, \ldots, m + (m-1)c$, where $(m-1)c + m < (m-1) \cdot (\frac{n-1}{m-1} - 1) + m = n$. Thus, as $h_c \leq n - 1$, we have $|L| \geq N_{h_c} + \mu_n \geq c + 3 = \lceil \frac{n-1}{m-1} \rceil + 1$.

We are left with the case where $\mu_1 = 0$ and $\mu_n = 0$.

**Case 5.** The number of columns is sufficiently large, that is, $n \geq 2m$. Let $c = \lfloor \frac{n}{m-1} \rfloor - 2$, where $c \geq 0$ as $n > m$. Consider columns $1, 2, \ldots, m - 1 + (m-1)c$,

where $m-1+(m-1)c = (m-1)(c+1) \leq (m-1)\cdot(\frac{n}{m-1}-1) = n-m+1$. By Claim 2, the last $m-1$ columns have at least two elements of $L$. Thus, $|L| \geq N_{h'_c} + 2 \geq c+4 = \lfloor \frac{n}{m-1} \rfloor + 2 \geq \lceil \frac{n}{m-1} \rceil + 1 \geq \lceil \frac{n-1}{m-1} \rceil + 1$.

We are left with the case where $n$ satisfies $3 \leq n \leq 2m-1$.

**Case 6.** The number of columns satisfies $m+2 \leq n \leq 2m-1$. Since $C_n$ has no elements of $L$, using Claim 3, $L$ is a CLS for the sub-mesh consisting of first $n-1 \geq m+1$ columns and $m$ rows, and by induction, $|L| \geq 3$.

**Case 7.** The number of columns satisfies $n = m+1$, and at least one row out of $R_1$ and $R_m$ has no element of $L$ (i.e., at least one of the following holds: $R_1 \cap L = \emptyset$, $R_m \cap L = \emptyset$). In this case, using Claim 3, $L$ is a CLS for a mesh of $m-1$ rows and $n$ columns. Since $n = (m-1)+2$, using induction we have $|L| \geq 3$.

We are left with the case where $n = m+1$, $|L \cap R_1| \geq 1$ and $|L \cap R_m| \geq 1$. Notice that the case where $|L| \geq 3$ satisfies the demand since $|L| \geq \lceil \frac{m}{m-1} \rceil + 1 = 3$.

The last case to consider is $n = m+1$, $|L| \leq 2$, so $|L \cap R_1| = 1$ and $|L \cap R_m| = 1$, and no other row has an element of $L$. We show that this scenario is not possible.

**Case 8.** Let $L = \{(1, y_1), (m, y_2)\}$. Since $C_1$ and $C_n$ have no elements of $L$, we have $2 \leq y_1, y_2 \leq n-1$. By Claim 2, $L$ has at least one element on columns $C_1, C_2, \ldots, C_{\lceil \frac{m}{2} \rceil}$ and symmetrically at least one element on columns $C_{\lfloor \frac{m}{2} \rfloor + 2}, \ldots, C_{m+1}$. As $\lceil \frac{m}{2} \rceil < \lfloor \frac{m}{2} \rfloor + 2$, we find that one of $y_1, y_2$ is in $\{1, 2, \ldots, \lceil \frac{m}{2} \rceil\}$, and the other is in $\{\lfloor \frac{m}{2} \rfloor + 2, \ldots, m+1\}$. Without loss of generality, by possibly reflecting the mesh across a vertical line, assume that $y_1 \leq \lceil \frac{m}{2} \rceil$ and $y_2 \geq \lfloor \frac{m}{2} \rfloor + 2$. Moreover, by possibly rotating the mesh by 180 degrees, we assume that $y_1$ is at least as far from the first column as $y_2$ is to the last column, that is, $y_1 - 1 \geq n - y_2 = m+1-y_2$. Consider the vertices $v_1 = (y_1 - 1, 1)$ and $v_2 = (y_1, 1)$. Note that $v_1$ is well defined since $y_1 \geq 2$. If $L$ is a CLS, these two vertices are separated by $(1, y_1)$ or by $(m, y_2)$. We have $d((1, y_1), v_1) = d(1, y_1), v_2) = y_1 - 1$, so $(1, y_1)$ does not separate $v_1$ and $v_2$. We also have $d((m, y_2), v_1) = \max\{m - y_1 + 1, y_2 - 1\}$ and $d((m, y_2), v_1) = \max\{m - y_1, y_2 - 1\}$. Since $y_2 - 1 \geq m - y_1 + 1 > m - y_1$, we get $d((m, y_2), v_1) = d((m, y_2), v_2) = y_2 - 1$, so $v_1$ and $v_2$ are not separated by a vertex of $L$, a contradiction. □

# 3 An upper bound

Given $M_{m,n}$ with $n \geq m \geq 2$, we define a set $L \subseteq V$ and show that its cardinality is according to Theorem 1 (i.e., $\lceil \frac{n-1}{m-1} \rceil + 1$), and that it is a landmark set (from now

on we only consider LS not CLS like in the previous section). Let $f = \lfloor \frac{n-1}{m-1} \rfloor + 1$. For $1 \leq k \leq f$, if $k$ is odd, let $z_k = (1, 1 + (k-1)(m-1))$, and if $k$ is even, let $z_k = (m, 1+(k-1)(m-1))$. Note that $1+(f-1)(m-1) \leq 1+\frac{n-1}{m-1} \cdot (m-1) = n$, so $z_1, z_2, \ldots, z_f$ are well-defined. That is, there is an element of $L$ every $m-1$ columns, starting with the first column. The rows of these elements alternate between 1 and $m$. If $1 + (f-1)(m-1) = n$, that is, $n-1$ is divisible by $m-1$, vertex $z_f$ is on column $n$, moreover, in this case, $\lceil \frac{n-1}{m-1} \rceil = \lfloor \frac{n-1}{m-1} \rfloor$, and we have defined $L$ completely. If $1 + (f-1)(m-1) < n$, that is, $n-1$ is not divisible by $m-1$, we have $\lceil \frac{n-1}{m-1} \rceil = \lfloor \frac{n-1}{m-1} \rfloor + 1$, and we define $z_{f+1} = (1, n)$ if $f$ is even (and $f+1$ is odd), and otherwise we let $z_{f+1} = (m, n)$.

Let $f' = \lceil \frac{n-1}{m-1} \rceil + 1$. The elements of $L$ are $z_1, \ldots, z_{f'}$ (no matter whether $f' > f$ or $f' = f$). Figure 1 shows an example of the set $L$ for a mesh with $m = 5$ and $n = 12$.

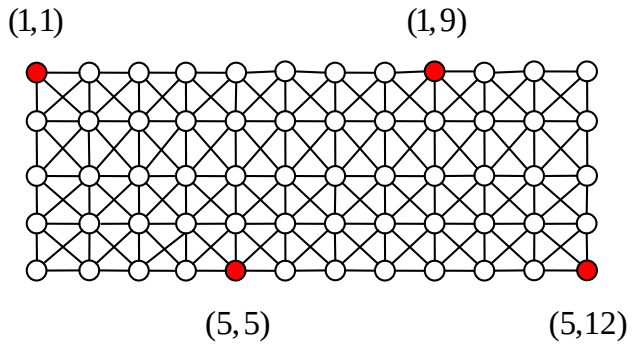



Figure 1: An example of the set $L = \{(1,1), (5,5), (1,9), (5,12)\}$ described in the above explanation, for a mesh with 5 rows and 12 columns.

**Claim 4.** *Consider the case $n \geq 2m-1$, and consider two vertices $v_1 = (x_1, y_1)$ and $v_2 = (x_2, y_2)$. If neither $z_1$ nor $z_{f'}$ separates $v_1$ and $v_2$, then $v_1$ and $v_2$ are on the same column (that is, $y_1 = y_2$).*

*Proof.* Assume by contradiction that $y_1 \neq y_2$ and assume without loss of generality that $y_1 < y_2$. If $y_2 \geq m+1$, we have $d(v_2, z_1) = \max\{x_2 - 1, y_2 - 1\}$. Since $x_2 - 1 \leq m-1$ while $y_2 - 1 \geq m$, we have $d(v_1, z_1) = d(v_2, z_1) = y_2 - 1$ where the first equality holds as $z_1$ does not separate $v_1$ and $v_2$. Next, by $y_2 - 1 = d(v_1, z_1)$ and $d(v_1, z_1) = \max\{x_1 - 1, y_1 - 1\}$, and since $x_1 - 1 \leq m-1$, we get $d(v_1, z_1) = y_1 - 1$. Thus $y_1 = y_2$ in this case, a contradiction.

If $y_2 \leq m$, we get $y_1 \leq m-1$, and the proof is symmetric for the distances from $z_2$ instead of $z_1$ (by rotating the mesh and possibly reflecting it across a horizontal

line); the distance of $v_1$ to $z_2$ is at least $m$, and since $v_2$ has the same distance, the maximum value of the distance is achieved by the second term also for $v_2$ and they are on one column. □

**Claim 5.** *Consider two vertices $u_1 = (1, 1)$ and $u_2 = (m, q)$, such that $2 \leq q \leq n$ and $q \leq m$. Consider also two distinct vertices $v_1 = (x_1, y)$ and $v_2 = (x_2, y)$ (on one column), where $1 \leq x_1 < x_2 \leq m$ and $1 \leq y \leq q$. Then, at least one of $u_1$ and $u_2$ separates $v_1$ and $v_2$.*

*Proof.* If $x_1 \geq y$, we have $x_2 \geq y + 1$. Thus, for $i = 1, 2$, $d(v_i, u_1) = \max\{x_i - 1, y - 1\} = x_i - 1$, and since $x_1 - 1 \neq x_2 - 1$, we find that $u_1$ separates $v_1$ and $v_2$. Otherwise, $x_1 < y$ holds, and we show that $u_2$ separates $v_1$ and $v_2$. So $d(v_1, u_2) = \max\{m - x_1, q - y\}$, and since $q \leq m$, $-y < -x_1$, we find $d(v_1, u_2) = m - x_1$. On the other hand, $d(v_2, u_2) = \max\{m - x_2, q - y\}$, where $m - x_2 < m - x_1$ (as $x_1 < x_2$) and $q - y < m - x_1$, so $d(v_2, u_2) = \max\{m - x_2, q - y\} < d(v_1, u_2)$, and therefore $u_2$ separates $v_1$ and $v_2$. □

**Corollary 1.** *Consider two vertices $u_1 = (1, 1)$ and $u_1 = (m, q)$, such that $2 \leq q \leq n$ and $q \geq m$. Consider two distinct vertices $v_1 = (x, y_2)$ and $v_2 = (x, y_2)$ (on one row), where $1 \leq y_1 < y_2 \leq q$ and $1 \leq x \leq m$. Then, at least one of $u_1$ and $u_2$ separates $v_1$ and $v_2$.*

The corollary holds by rotating the mesh by 90 degrees.

**Claim 6.** *Consider two distinct vertices $v_1 = (x_1, y)$ and $v_2 = (x_2, y)$ (on one column). Let $z_s = (x_s, y_s)$ and $z_{s+1} = (x_{s+1}, y_{s+1})$ be such that $y_s \leq y \leq y_{s+1}$. Then, at least one of $z_s$ and $z_{s+1}$ separates $v_1$ and $v_2$.*

*Proof.* Without loss of generality it is sufficient to consider the sub-graph of the mesh consisting of columns $y_s, y_s + 1, \ldots, y_{s+1}$ and all rows. The property holds by Claim 5, as no shortest path between two vertices in this sub-mesh traverses any vertex outside it, and by possibly reflecting the mesh across a vertical line. □

**Corollary 2.** *In the case $n \geq 2m - 1$, the set $L$ is a landmark set.*

*Proof.* By Claim 4, for every pair of vertices on different columns, $L$ contains a vertex separating them. By Claim 6, every pair of vertices on one column is separated as well, by choosing an appropriate value of $s$, which is possible for any $y$ since $L$ has an element on $C_1$ and an element on $C_n$. □

We are left with the case where $n$ satisfies $m + 1 \leq n \leq 2m - 2$ (in particular, $n \geq 3$). In this case, $L = \{z_1 = (1, 1), z_2 = (m, m), z_3 = (1, n)\}$.

**Claim 7.** *Consider two distinct vertices $v_1 = (x_1, y_1)$ and $v_2 = (x_2, y_2)$, in the case where $n \in \{m+1, m+2, \ldots, 2m-2\}$. At least one of the vertices $z_1$ and $z_3$ separates $v_1$ and $v_2$.*

*Proof.* Assume that $z_1$ does not separate $v_1$ and $v_2$. We have $d(v_i, z_1) = \max\{x_i - 1, y_i - 1\}$.

If $x_1 = x_2$, we have $y_1 \neq y_2$. By $\max\{x_1-1, y_1-1\} = \max\{x_2-1, y_2-1\}$ we find that it cannot be the case that $d(v_i, z_1) = y_i - 1$ for $i = 1, 2$. Thus, we can assume without loss of generality that $d(v_1, z_1) = x_1 - 1 > y_1 - 1$. We claim that $x_2 \geq y_2$ holds. Indeed, $d(v_2, z_1) = d(v_1, z_1) = x_1 - 1 = x_2 - 1$, so $x_2 - 1 \geq y_2 - 1$. Thus, as $x_1 = x_2 \leq m$, we have $y_1 < m$ and $y_2 \leq m$, and by Corollary 1, $z_2 = (m, m)$ separates $v_1$ and $v_2$.

If $y_1 = y_2$, by Claim 6, one of $z_1$, $z_2$, and $z_3$ separates $v_1$ and $v_2$.

We are left with the case $x_1 \neq x_2$ and $y_1 \neq y_2$. By the definition of distances, we cannot have that $d(v_i, z_1) = x_i - 1$ holds for $i = 1, 2$ or that $d(v_i, z_1) = y_i - 1$ holds for $i = 1, 2$, so we have either $d(v_1, z_1) = y_1 - 1$ and $d(v_2, z_1) = x_2 - 1$ (and $x_1 < y_1$, $x_2 > y_2$) or we have $d(v_1, z_1) = x_1 - 1$ and $d(v_2, z_1) = y_2 - 1$ (and $x_1 > y_1$, $x_2 < y_2$). Without loss of generality (by possibly swapping the roles of $v_1$ and $v_2$) we assume that $d(v_1, z_1) = y_1 - 1$ and $d(v_2, z_1) = x_2 - 1$ holds, so $y_1 = x_2$, and $x_1 < y_1$, $x_2 > y_2$.

Consider the distances to $z_3$. We find $d(v_1, z_3) = \max\{x_1 - 1, n - y_1\}$ and $d(v_2, z_3) = \max\{x_2 - 1, n - y_2\}$. Assume that $d(v_1, z_3) = x_1 - 1$ holds. If $d(v_2, z_3) = x_2 - 1$, we are done by $x_1 \neq x_2$. Otherwise $d(v_2, z_3) = n - y_2$, and we have $x_1 - 1 > n - y_1$ and $x_2 - 1 < n - y_2$. We get $n - y_2 > x_2 - 1 = y_1 - 1 > x_1 - 1$. Assume that $d(v_1, z_3) = n - y_1$ holds. If $d(v_2, z_3) = n - y_2$, we are done by $y_1 \neq y_2$. Otherwise $d(v_2, z_3) = x_2 - 1$, and we have $x_1 - 1 < n - y_1$ and $x_2 - 1 > n - y_2$. We get $x_2 - 1 > n - y_2 > n - x_2 = n - y_1$, so $z_3$ separates $v_1$ and $v_2$ in all remaining cases. □

# 4 Conclusion

In section 2 we have proved a lower bound of $\lceil \frac{n-1}{m-1} \rceil + 1$ on the column metric dimension of $M_{m,n}$, for all values of $m, n \geq 2$, implying the lower bound for the metric dimension for $n > m \geq 2$. In section 3 we have proved an upper bound of $\lceil \frac{n-1}{m-1} \rceil + 1$ on the metric dimension of $M_{m,n}$, for all values of $n > m \geq 2$, by defining a suitable landmark set. As mentioned in Section 1, the case of $m = n$ is a special case where $MD(M_{m,n}) = 3$ [15], which was known prior to our work, and we do not analyze this case. Our main result (Theorem 1) is proved by combining the lower bound and upper bound. Using the last remark we get a full characterization of the metric dimension of (two-dimensional) meshes.

## References

[1] R. Adar, L. Epstein. An algorithm for the weighted metric dimension of two-dimensional grids. arXiv:1602.05899

[2] P. Andersen, C. Grigorious, and M. Miller. Minimum weight resolving sets of grid graphs. *Discrete Mathematics, Algorithms and Applications*, 8(3), 22 pages.

[3] L. Babai. On the order of uniprimitive permutation groups. *Annals of Mathematics*, 113(3):553–568, 1981.

[4] Z. Beerliova, F. Eberhard, T. Erlebach, A. Hall, M. Hoffmann, M. Mihalák, and L. S. Ram. Network discovery and verification. *IEEE Journal on Selected Areas in Communications*, 24(12):2168–2181, 2006.

[5] J. Cáceres, M. C. Hernando, M. Mora, I. M. Pelayo, M. L. Puertas, C. Seara, and D. R. Wood. On the metric dimension of cartesian products of graphs. *SIAM Journal on Discrete Mathematics*, 21(2):423–441, 2007.

[6] G. Chartrand, L. Eroh, M. A. Johnson, and O. R. Oellermann. Resolvability in graphs and the metric dimension of a graph. *Discrete Applied Mathematics*, 105(1-3):99–113, 2000.

[7] G. Chartrand and P. Zhang. The theory and applications of resolvability in graphs: A survey. *Congressus Numerantium*, 160:47–68, 2003.

[8] V. Chvátal. Mastermind. *Combinatorica*, 3(3):325–329, 1983.

[9] J. Díaz, O. Pottonen, M. J. Serna, and E. J. van Leeuwen. Complexity of metric dimension on planar graphs. *Journal of Computer and System Sciences* 83(1):132-158, 2017.

[10] L. Epstein, A. Levin, and G. J. Woeginger. The (weighted) metric dimension of graphs: Hard and easy cases. *Algorithmica*, 72(4):1130-1171, 2015.

[11] F. Harary and R. Melter. The metric dimension of a graph. *Ars Combinatoria*, 2:191–195, 1976.

[12] M. Hauptmann, R. Schmied, and C. Viehmann. Approximation complexity of metric dimension problem. *Journal of Discrete Algorithms*, 14:214–222, 2012.

[13] V. Jude Annie Cynthia. Metric dimension of certain mesh derived graphs. *Journal of Mathematical and Computational Science* Vol.5 (1), 71-77, 2014

[14] S. Khuller, B. Raghavachari, and A. Rosenfeld. Landmarks in graphs. *Discrete Applied Mathematics*, 70(3):217–229, 1996.

[15] R. A. Melter and I. Tomescu. Metric bases in digital geometry. *Computer Vision, Graphics, and Image Processing*, 25:113–121, 1984.

[16] A. Sebö and E. Tannier. On metric generators of graphs. *Mathematics of Operations Research*, 29(2):383–393, 2004.

[17] B. Shanmukha, B. Sooryanarayana, and K. S. Harinath. Metric dimension of wheels. *Far East Journal of Applied Mathematics*, 8(3):217–229, 2002.

[18] P. J. Slater. Leaves of trees. *Congressus Numerantium*, 14:549–559, 1975.

*Received 1st February 2018*

Acta Cybernetica **23** (2018) 773–782.

# Regional Multicriteria and Multimodal Route Planning System for Public Transportation: A Case Study*

József Békési[a]

**Abstract**

Nowadays, the use of computer-based route planners is popular among private and public transportation passengers. A large range of websites and GPS navigation devices provide such services for their users. Here, we present an algorithm used by a route planning system which operates on a complete public transport network of two regions from two countries, namely Hungary and Serbia. The algorithm can handle the pedestrian traffic between stops not too far from each other. It can take into account individual user preferences like walking distances and modes of transport. The graph representing the transport network was very large, but with the help of some speed-up techniques, we managed to create an effective search algorithm that is able to handle user requirements.

**Keywords:** public transportation, route planning, algorithms

## 1 Introduction

Nowadays, the use of computer-based route planners is widespread among passengers. Individual passengers have a large choice of websites and GPS navigation devices. Generally speaking, with these kind of systems the available road network is modeled by a graph. The vertices of the graph represent the points of contact of the roads, while the edges represent the road sections connecting the points. If we assign the length of the road section to the edges, we get a weighted graph. We can use the well-known Dijkstra algorithm to calculate the shortest distance between two points. The efficiency of the Dijkstra algorithm is quite good, but the size of the graphs describing real-world road networks can be very large, especially in the case of a larger geographical area like the road network of a continent. As passengers generally expect an almost immediate response to their searches, even the Dijkstra

*This research work was supported by the EU-funded Hungarian grant EFOP-3.6.2-16-2017-00015.

[a]Department of Applied Informatics, Gyula Juhász Faculty of Education, University of Szeged, E-mail: bekesi@jgypk.szte.hu

DOI: 10.14232/actacyb.23.3.2018.3

algorithm with polynomial running time is not good enough for large graphs. Over the last two decades, many potential speed-ups have been investigated to address the problem. More details about this can be found in [6].

Routing services are available not only for private passengers, but for passengers travelling by public transportation as well. In this case, usually the road network does not need to be described by the graph. The reason for this is that the route is pre-determined, or the journey is often not on a conventional road network but on a bound track (e.g. rail) or in the air, or even on water. The nodes of public transport routing graphs generally represent the vehicle stops, and the edges provide travel possibilities between stops. The weight of the edges may represent the travel time, but as different users may look at travel from different aspects, other models should be considered. Public transport networks may generally be larger than those of road networks, given that travel is time-dependent, and handling this is also required in the model. Therefore, in this case, the opportunities for speeding-up searches are particularly important, and they have also been widely investigated. Some of the speed-up options available on road graphs can be used here, but not those that use the special features of road networks. For more details on models and speed-up options, see [4, 11].

In practice, it is usual for public transportation companies to provide route planners for their own service area. This usually includes the services of the company or other companies closely related to that city, region or country. In most cases, however, these services provide only route planning for one kind of journey; for example, a route planner of a rail company can be searched for rail routes, while route planners of bus companies can be used for bus routes. In the case of local transport, it is common that there are search engines that cover different modes of transport in a given city, such as buses, trams and metro, trains, but they are usually not intended for long-distance transport. So, for instance, if a passenger wants to get from a point in a city to a point in another city, using local and then several long-distance modes of transport, he or she usually will not find a search engine that offers such a travel option, not even in a country or in a region. And the graphs of road networks can handle the same problem for larger areas such as continents. This suggests that finding a route to public transport is a more difficult issue than planning private trips.

In this paper, we present a route planning system and its search algorithm, where the latter operates on a complete public transport network of two regions from two countries, namely Hungary and Serbia. The databases on which the model is based include long-distance trains, buses and complete local transport of the major cities. The databases are based on timetables taken from service providers operating in the regions. The system can also model the pedestrian traffic between stops not too far from each other. It can take into account individual user preferences, like walking distances, modes of transport, and properties of the objective function. The graph representing the transport network was very large, but with the help of some speed-up techniques, we managed to create an effective search algorithm that is able to handle user requirements.

The algorithm was developed under the EU-funded Hungary-Serbia Interreg-

IPA CBC Program as part of a complex route planning application. This study presents the most common methods used in the search algorithm, as well as the experiences and results obtained during the practical running.

## 2 Preliminaries and literature review

Next, we will summarize the key results of public transport route algorithms described in the literature. We will mainly refer to the technology used by the method we use, and in the following discuss the topic in more detail [6, 12].

The simplest modelling of public transport networks can be achieved using the so-called Station Graph [15]. In this case, the graph's nodes represent the stops and edges only exist between two stops if one of the stops can be reached directly from the other. Generally speaking, the Station Graph is a relatively small graph, since each stop has exactly one node and there is at most one edge between any two stops. However, this graph contains limited information as it does not represent the time of the routes at all. Hence it is not suitable for precise route planning, and it can only provide a poor estimate for the length of the trip or for the minimum required transfers.

The Time-Dependent model may be regarded as an extension of the Station Graph [5], in which the edges may have multiple weights. During a route search, the current weight is calculated using the given time, taking into account the bus lines departing from the stop. In this case, the size of the graph does not increase, but the determination of the weight requires more complex calculations, which may slow down the search. The model has often been studied on railway networks and has been developed, for instance, for the correct modelling of transfers [13].

The most common model used is the Time-Expanded model [14]. Here, the starting and arrival times of the vehicles are represented by special nodes. The nodes belonging to a station can be sorted by time and the waiting can be represented by edges. Trips between the different stations can be expressed by edges between the appropriate departure and arrival times. Therefore the model can handle transfers, and pedestrian traffic between two specific stops can also be modelled. Initially, due to the large size of the graph, searches in this model were not sufficiently fast, but due to technological developments and improvements, we can now consider it a competitive method [6].

While the aim of search engines is normally to find the shortest route between departures and arrivals, this is not always the case for public transport networks. Passengers may wiew things from different aspects; some may want to minimize the number of passes against the earliest arrival, while others may want to keep their travel costs as low as possible. What is more, we usually choose one mode of transport on the route; for example, like that for a motorist or pedestrian. With public transport, someone may want to use a variety of vehicles or does not even have the opportunity to reach the goal otherwise; for example, he or she needs to combine buses and railways, and has to reach the stop on foot. Hence special models have also been developed for public transport networks. This is why besides

the classical earliest arrival problem, [14] multiobjective optimization methods were applied [11].

The research work of recent years has focused mainly on developing complex multiobjective, multimodal route-planning algorithms [3, 9], which include appropriate speed-up techniques [8] and they can be applied in practice [1].

# 3 Modeling

All three types of graphs described above were used to model our transport network. Only the Station Graph and the Time-Dependent Ggraph built on it were stored permanently in the memory. The number of edges of the Time-Dependent Graph was already quite large as there were seven different timetable versions in use during that period. This meant that searches for different days and periods were subject to different schedules. For example, the Sunday schedule was different from the weekday one. In the Time-Dependent Graph there were different edges for the different lines, and the departure and arrival times of the lines were stored in separate structures. It also included possible walking routes. And the Station graph and the Time Dependent graph did not have any direct search, but only had roles in the preprocessing of searches. In the actual search, the Dijkstra algorithm was implemented on a Time Expanded Graph. The current Time Expanded Graph was not stored permanently in memory as the many different timetable versions dynamically changed it. The generation of the current graph was always related to the actual query.

Table 1: Sizes of the graphs

| Graph type | Number of nodes | Number of edges |
|---|---|---|
| Station | 12014 | 416163 |
| Time Dependent | 12014 | 6786662 |
| Time Expanded | $\approx 562000$ | $\approx 4616000$ |

Table 1 lists the sizes of different graphs.
Figure 1 depicts the modelling of a line with the Station Graph.

Figure 2 shows a part of the Time Expanded Graph. The horizontally positioned nodes are part of a stop's timeline, and the pink nodes represent the departure times, the green ones represent the arrival times. The orange edges are the waiting edges of the timeline. The black edges model the lines between the stops, while the red edges show the walking options. The departure timeline is in the upper line, the arrival is in the lowest one. Thickly marked paths indicate the travel possibilities chosen by the system.

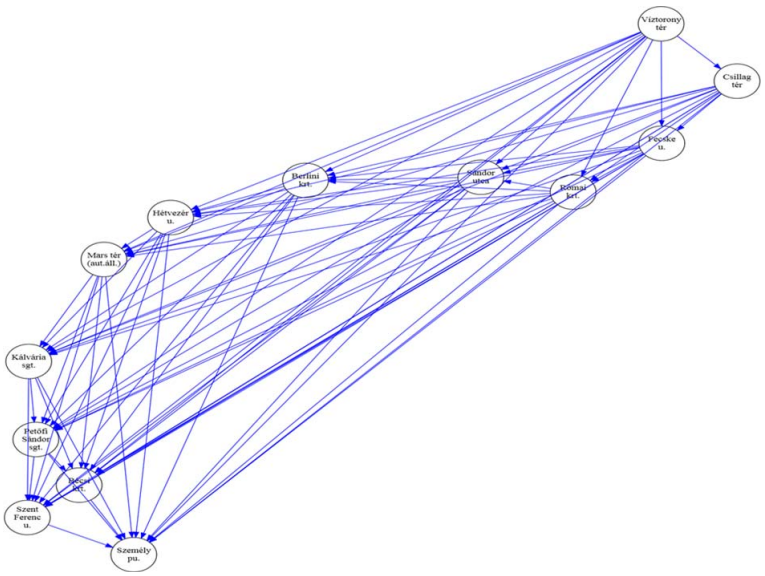



Figure 1: The Station Graph of a line

# 4 Search algorithm

After producing the Time Expanded Graph and weighing the edges appropriately, with the help of the Dijsktra algorithm we can find the shortest path. In our case, we came up against a difficulty caused by the search dependence of the Time Expanded Graph. Its structure was influenced by the time specified in the search and by the modes of transport that the user wanted to use, or by other parameters such as the maximum allowed walking distance. Despite the large size of the graph, its generation was relatively fast, but after including the search time, in some cases we could not achieve the desired speed with this method. In addition, the structure of the entire graph resulted in excessive memory usage during a search, which was a real problem, because multiple clients wanted to use the services of the server simultaneously.

To overcome the problems listed above, we performed a preprocessing step before producing the Time Expanded Graph and the search. The aim was to determine the nodes which include some of the shortest paths corresponding to the user parameters and the objective function. The method is similar to the TRANSIT algorithms described in the literature [1, 2, 9]. In the end, the Time Expanded Graph contained far fewer nodes, and it led to a significant speed-up of the search.

The purpose of the preprocessing was to determine the nodes that lie on the shortest paths from the source to the target. Here, we used the Station Graph for the prepocessing part. Our aim was to determine all the nodes that lie on the maximum k-length paths between the source and destination. In this case, the

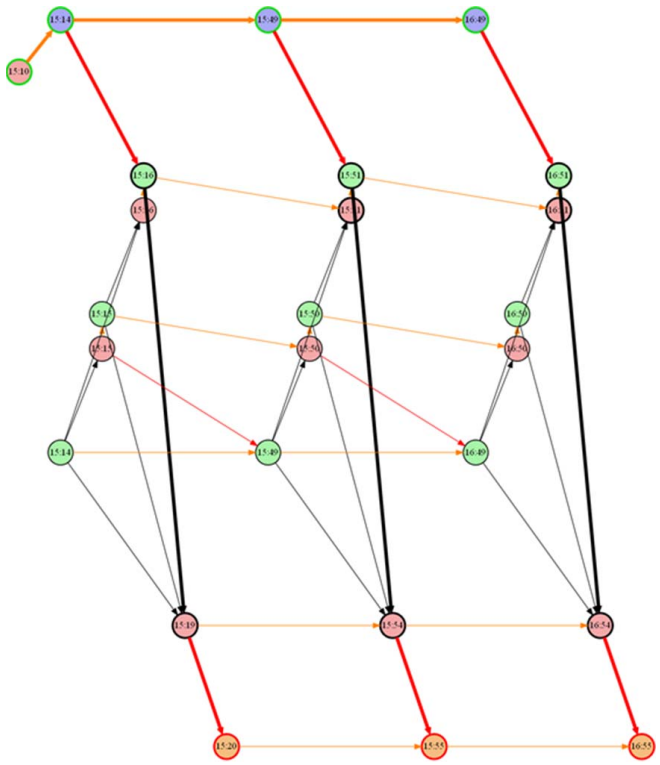


Figure 2: Part of the Time Expanded Graph

parameter k means the maximum number of transfers that can be made during the journey. Here, we used a modified version of the depth first search algorithm that did not take into account the nodes farther from the source than the desired length. Given that in some cases the execution of the procedure might take longer, the value of the parameter k and the execution time were also limited in the search. Experience has shown that these heuristics worked well in practice so, in almost every case, the Time Expanded Graph built on these nodes contained the shortest path of the entire graph. For a more detailed analysis of this, see Section 6.

# 5 The objective function

In general, different objective functions are used for public transport route planning. One of the most common is the earliest arrival, but it does not always fully meet the user's preferences. Finding different Pareto optima is also a common problem. We used an objective function that included a weighted sum of different goals. And the weights were determined based on information provided by the users, using preset patterns.

The weight of a travel edge is the following: $t_r w_r + f_r$, where $t_r$ is the time of travel, $w_r$ is the actual weight of the journeys, and $f_r$ is the additive factor for the travel cost.

The weight of a waiting edge is the following: $t_i w_i$, where $t_i$ is the waiting time, and $w_i$ is the actual weight of the waiting.

The weight of a pedestrian edge is the following: $t_a w_a + f_a$, where $t_a$ is the time of walking, $w_a$ is the actual weight of the walking, and $f_a$ is the additive factor for pedestrian cost.

In the search, there were three options available to the users regarding the search query. These three options are the fastest route, the minimum number of connections and the minimum walking distance. And the parameters of the weights for each goal are given in Table 2.

Table 2: Parameters of the weights

| Options | $w_r$ | $f_r$ | $w_i$ | $w_a$ | $f_a$ |
|---|---|---|---|---|---|
| Fastest | 1 | 30 | 0.8 | 1.5 | 30 |
| Less transfer | 1 | 1000 | 0.8 | 1.5 | 10 |
| Less walking | 1 | 30 | 0.8 | 30 | 1000 |

# 6 Results and analysis

We investigated the proportion of the investigated instances such that the reduced graph contained all nodes of the shortest path. This ensured that the search on the reduced Time Expanded Graph would give an optimal solution. In the preprocessing heuristics we modified two parameters. One was the previously mentioned k parameter, while the other was the search time. For the possible values of k, we selected s+1 and s+2, where s is the shortest path between two points in the Station Graph.

Here, we used two types of test queries. They both contained 1000 queries, one of which concerned questions regarding local traffic (T1) and the other concerned long-distance traffic (T2). In the table below, we summarize how efficient the preprocessing heuristics was for the two types of test data, for different values of k. In the table out of the 1000 questions we see how many times the reduced-size Time Expanded Graph gave a solution that differed from the optimum, which we calculated using the complete Time Expanded Graph.

Tables 4 and 5 contain the average and maximum search times in milliseconds on the above-mentioned query lists for the reduced and for the complete Time Expanded Graphs. The notations used for the columns are the following:
RB: Reduced graph build time
RS: Reduced graph search time
RC (=RB+RS): Reduced graph complete search time

Table 3: Non-optimality statistics of the prepocessing heuristics

| Input type | $k = s + 1$ | $k = s + 2$ |
|---|---|---|
| T1 | 13 | 0 |
| T2 | 54 | 11 |

FB: Full graph build time
FS: Full graph search time
FC (=FB+FS): Full graph complete search time

Table 4: Average running times in milliseconds

| | RB | RS | RC | FB | FS | FC |
|---|---|---|---|---|---|---|
| $T1, k = s + 1$ | 308 | 18 | 327 | 5327 | 166 | 5493 |
| $T1, k = s + 2$ | 1394 | 406 | 1801 | | | |
| $T2, k = s + 1$ | 256 | 55 | 312 | 5427 | 1137 | 6564 |
| $T2, k = s + 2$ | 1186 | 316 | 1502 | | | |

Table 5: Maximal running times in milliseconds

| | RB | RS | RC | FB | FS | FC |
|---|---|---|---|---|---|---|
| $T1, k = s + 1$ | 1478 | 446 | 1529 | 5744 | 4130 | 9502 |
| $T1, k = s + 2$ | 4959 | 2458 | 5453 | | | |
| $T2, k = s + 1$ | 2491 | 788 | 3279 | 6290 | 10216 | 15650 |
| $T2, k = s + 2$ | 3119 | 3938 | 6469 | | | |

# 7 Conclusions

We presented an algorithm used by a route planning system for a complete public transport network of two regions from Hungary and Serbia. The graph representing the transport network was very large, but with the help of some speed-up techniques, we managed to create an effective search algorithm that is able to handle user requirements.

## 8 Acknowledgments

This study was carried out in cooperation with The Public City Transport Enterprise of Novi Sad, Department of Applied Informatics, University of Szeged, University of Novi Sad, DAKK Ltd., Szeged, Newline Ltd., Szeged and TIAC Ltd., Novi Sad.

## References

[1] L. Antsfeld, T. Walsh, Finding Optimal Paths in Multi-modal Public Transportation Networks using Hub Nodes and TRANSIT algorithm, In.: L. Frommberger,K. Schill, B. Scholz-Reiter (eds.), Proceedings of the 3rd Workshop on Artifical Intelligence and Logistics, Montpellier, France, 7-13, 2012.

[2] H. Bast, S. Funke, D. Matijevic. Transit ultrafast shortest-path queries with linear-time preprocessing. In 9th DIMACS Implementation Challenge, 2006.

[3] H. Bast, E. Carlsson, A. Eigenwillig, R. Geisberger, C. Harrelson, V. Raychev, F. Viger, Fast routing in very large public transportation networks using transfer patterns, Proceedings of the 18th Annual European Symposium on Algorithms (ESA'10), Lecture Notes in Computer Science, Vol. 6346, Springer, 290-301, 2010.

[4] H. Bast et al., Route Planning in Transportation Networks. In: L. Kliemann, P. Sanders (eds) Algorithm Engineering. Lecture Notes in Computer Science, Vol. 9220. Springer, Cham, 2016.

[5] G. Brodal, R. Jacob. Time-dependent Networks as Models to Achieve Fast Exact Time-table Queries. Electronic Notes in Theoretical Computer Science, Vol. 92, 3-15, 2004.

[6] D. Delling, Engineering and augmenting route planning algorithms, Ph.D. thesis, Universität Karlsruhe, Fakultät für Informatik, 2009.

[7] D. Delling, T. Pajor, D. Wagner, Engineering time-expanded graphs for faster timetable information, Robust and Online Large-Scale Optimization, Lecture Notes in Computer Science, Vol. 5868, Springer, 182-206, 2009.

[8] D. Delling, T. Pajor, D. Wagner, Accelerating multimodal route planning by access-nodes, Proceedings of the 17th Annual European Symposium on Algorithms (ESA'09), Lecture Notes in Computer Science, Vol. 5757, Springer, 587-598, 2009.

[9] J. Koszelew, Two Methods of Quasi-Optimal Routes Generation in Public Transportation Network, International Conference on Computer Information Systems and Industrial Management Applications,231-236, 2008,

[10] M. Müller-Hannemann, K. Weihe, Pareto shortest paths are often feasible in practice, Proceedings of the 5th International Workshop on Algorithm Engineering (WAE'01), Lecture Notes in Computer Science, Vol. 2141, Springer, 185-197, 2001

[11] M. Müller-Hannemann, F. Schulz, D. Wagner, C. Zaroliagis, Timetable Information: Models and Algorithms. In: F. Geraets, L. Kroon, A. Schoebel, D. Wagner, C.D. Zaroliagis (eds) Algorithmic Methods for Railway Optimization. Lecture Notes in Computer Science, Vol. 4359. Springer, Berlin, Heidelberg, 2007

[12] T. Pajor, Algorithm Engineering for Realistic Journey Planning in Transportation Networks, Ph.D.thesis, Universität Karlsruhe, Fakultät für Informatik, 2013.

[13] E. Pyrga, F. Schulz, D. Wagner, C. Zaroliagis, Efficient models for timetable information in public transportation systems, *ACM Journal of Experimental Algorithmics* Vol. 12 No. 2.4, 1-39., 2008

[14] F. Schulz, D. Wagner, K. Weihe, Dijkstra's Algorithm On-Line: An Empirical Case Study from Public Railroad Transport, *ACM Journal of Experimental Algorithmics*, Vol. 5, No. 12, 2000.

[15] F. Schulz, D. Wagner, C. Zaroliagis, Using multi-level graphs for timetable information in railway systems, Proceedings of the 4th Workshop on Algorithm Engineering and Experiments (ALENEX'02), Lecture Notes in Computer Science, Vol. 2409, Springer, 43-59, 2002.

*Received 4th April 2018*

Acta Cybernetica **23** (2018) 783–800.

# Integrated Vehicle Scheduling and Vehicle Assignment

József Békési[a], Balázs Dávid[a], and Miklós Krész[a]

**Abstract**

The vehicle scheduling problem has been extensively studied in the past decades. Yet, most models and methods given in the literature consider only a theoretical scenario where vehicles just have to service the timetabled trips of the input. However, schedules created this way cannot be used in real life, as they should also consider constraints such as refueling, parking, and maintenance, which are all connected to the vehicle servicing the trips. In this paper, we give a set partitioning model for the multi-depot integrated vehicle scheduling and vehicle assignment problem. This model can also be used as a general framework, which can integrate multiple activities based on the rules or regulation of the different possible input scenarios. We give a column generation-based solution method, and demonstrate its efficiency on randomly generated test instances, which treat the refueling of vehicles with two different fuel types as the vehicle-specific activity.

**Keywords:** vehicle scheduling, vehicle assignment, integrated model, alternative fuel, refueling, maintenance

## 1 Introduction

The vehicle scheduling problem (VSP) is one of the most basic optimization problems arising in public transportation. It has been fairly well researched over the past few decades, but state-of-the-art models and methods mostly treat the problem as a mathematical one. The usual models and solution methods given for the VSP are interesting from a theoretical point of view, but many of the solutions cannot be applied directly in real life. They mainly focus on assigning timetabled trips to the vehicles of the company. However, there are other constraints that need to be considered while creating a vehicle schedule.

The vehicle schedules of a transportation company are not just virtual sets of tasks that have to be executed in the given sequence, but they also have a real vehicle (or at least a vehicle brand/type) assigned to the schedule as well. Knowing

[a]University of Szeged, Department of Applied Informatics, E-mail: {bekesi,davidb,kresz}@jgypk.szte.hu

DOI: 10.14232/actacyb.23.3.2018.4

the vehicle responsible for the execution of the trips also means that there are special needs that have to be taken into account while the vehicle is in service. For instance, it can run out of fuel, and has to be refueled, or spends too much time in service, and has to be sent for short maintenance during the day. While similar constraints are important from the perspective of a real-life application, they are not widely studied in the literature. We will refer to these requirements as vehicle-specific activities. More application-oriented approaches for the VSP started appearing with the rise of electric and alternative-fuel vehicles. These are both cheap and environmentally friendly to operate. However, their major drawback is that they can only run a limited distance, so refueling events have to be considered when creating their schedules [19].

While refueling is an important constraint that has to be included in a VSP model, other vehicle-specific activities should also be considered, such as parking and maintenance [15, 7]. Constraints like these receive significantly less attention than refueling. In this paper, we present a set partitioning model for the integrated vehicle scheduling and assignment problem with vehicle-specific activities. If such activities are included in a vehicle schedule, they also determine different tasks that the vehicle has to execute besides its timetable trips. This results in a vehicle assignment combined with scheduling. Our goal is to give a general framework where most vehicle-specific activities can be integrated, and provide a flexible model where many of these application-oriented constraints can be readily included. While there may exist other models and methods that deal with these problems separately, to our knowledge such activities have not been considered together in the same problem before.

We give a column generation-based solution method for this model, and demonstrate its efficiency on randomly generated test instances. To showcase the model, we use refueling as the vehicle-specific activity for these instances, and also study the concept of multiple fuel types, which is also rarely considered in the literature.

## 2 Vehicle scheduling and vehicle-specific activities

For the introduction of the VSP, we will follow our terminology from [10]. The input of the VSP is a set $V$ of vehicles and set $T$ of *timetabled trips*. Every trip has a departure and arrival time, a starting and ending location, and the distance that they cover. Trips also have a set of vehicles that are able to service them. A $(t, t')$ pair of trips are compatible if a vehicle can service both trips with respect to the running time and distance between the arrival location of $t$ and the departure location of $t'$. Traveling between two locations without servicing a timetabled trip is called a *deadhead trip*. Deadheads also have a starting and ending location, distance and duration in time.

The VSP gives an assignment between the timetabled trips and the vehicles in such a way that every trip is executed exactly once, and trips assigned to the same vehicle are pairwise compatible. As mentioned before, trips can determine the set of vehicles that can execute them, and this is achieved through the concept

of *depots*. Vehicles may belong to depots. These are determined by the features of the trips; namely, certain trips might be served only by vehicles that satisfy given constraints (e.g. is the vehicle wheelchair accessible, does the vehicle have air conditioning, or a given passenger capacity?). Such constraints are important, and are centered around the services you want to provide to passengers taking the given trip (e.g. people with a wheelchair should have no problem getting on any bus along the route of this trip), or a regulation connected to the trip (e.g. every bus covering a trip that is longer than a given distance must have air conditioning),

These depots are usually determined by a combination of two features; namely the type of vehicle (e.g. with/without air conditioning, solo, or articulated), and its starting location at the beginning of the day. When we talk about vehicles belonging to the same depot in this paper, we mean vehicles sharing the same vehicle type and having the same starting/ending geographical location at the beginning/end of the day. If the problem has at least two depots, every trip is also assigned a depot-compatibility vector that corresponds to the depots that can execute it.

Besides depots, *vehicle characteristics* will also be addressed. These also represent different attributes of the vehicles, but only those that do not have an influence on servicing trips. For example, the fuel type of the vehicle may be such a characteristic; vehicles belonging to the same depot can run on different fuels, but can still service the same trips.

The total cost of the problem usually consists of a one-time daily cost for each vehicle in service, and an operational cost proportional to the distance traveled by these vehicles. Both of these costs may be different depending on the depot of the vehicle. The result of the VSP is a daily vehicle schedule, which consists of vehicle blocks. A vehicle block is basically a sequence of tasks that are executed by the same vehicle.

If the problem has only one depot, it is called the single depot vehicle scheduling problem (SDVSP), and it can be solved in polynomial time. A formulation for the SDVSP is presented in [4]. A problem with at least two depots is referred to as a multiple depot vehicle scheduling problem (MDVSP). The MDVSP was introduced by Bodin et al. in [5], and its NP-hardness was shown by Bertossi et al. [3]. There are several different models and solution methods available for the problem, and Bunte et al. provide a good overview of these in [8].

Solution methods based on the basic concepts of the SDVSP and MDVSP cannot be applied directly in practice, as they only deal with covering all timetabled trips. In real life, however, vehicles have to execute several types of activities during the day. These come from different vehicle-specific needs (parking, refueling, maintenance, etc.), and they usually have to be executed after a vehicle has covered a certain distance (refueling, maintenance), or spent a certain amount of time without performing any activities (parking). In all of these cases, the total length of certain consecutive activities is limited, and vehicle-specific events have to be scheduled after such work pieces.

In general, extensions of the VSP that have either a time or distance constraint on the length of the vehicle blocks are called the Vehicle Scheduling Problem with Time/Route Constraints [13, 8]. These alone cannot satisfy the constraints for

vehicle-specific activities, as they limit the total length of the flow representing a block, while activities only need a limit for certain parts of this flow. However, most problems that consider the above vehicle-specific activities are special cases of this group.

In this paper, we introduce a set partitioning-based model for the integrated vehicle scheduling problem with vehicle-specific activities. The purpose of this model is to provide a general framework capable of producing vehicle schedules that can be used in practice. Most papers dealing with the VSP do not consider vehicle-specific activities, although they are really important real-life constraints for vehicles. One such activity that has gained increasing popularity in recent years is the scheduling of alternative fuel (natural gas, hybrid) or electric vehicles.

Vehicle scheduling for alternative fuel vehicles (AF-VSP) is hard because of the limited distance they can cover. Vehicles have to be refueled during their daily blocks, and there are usually very few special refueling stations, with a limited number of refueling pumps. Because of this, location problems for refueling stations are also important [17]. Li presented a flow network-based model for both alternative fuel and electric vehicles in [18]. Here, a single depot VSP is considered with a single fuel type, and a single refueling station at the depot. Several column generation-based solution techniques are presented on instances with up to 947 trips. Adler studied the problem with multiple depots in [1, 2]. He used a set partitioning model for alternative fuel vehicle scheduling, and also used column generation to solve instances with up to 50 trips. Larger instances are solved by applying heuristic methods.

Electric vehicle scheduling (E-VSP) has also been getting a lot of attention recently. Both the above-mentioned paper by Li [18] and the dissertation of Adler [1] present models and solution methods for the E-VSP. Their solution approaches are similar to those that were used for the AF-VSP. They mainly deal with battery swapping, and their solution approaches are also based on column generation. A time-space network-based model that allows the charging of the vehicles is given in Reuer et al. [23]. In [25], multiple models are presented for the E-VSP that consider battery charge. The solution is once again obtained using a column generation approach.

Another important vehicle-specific activity is the assignment of small maintenance tasks to vehicles during their daily blocks. According to Haghani and Shafahi [15], they can be of three types: daily, preventive and emergency maintenance. Daily inspections can be built into a vehicle schedule at the beginning or the end of the blocks, while emergency maintenance is only issued as part of a disruption management process when something unexpected happens to a vehicle. However, preventive maintenance has to be executed by vehicles after a set distance or time interval. Such maintenance activities were examined for rolling stock rotations by Borndörfer et al. in [7], while Haghani et al. also gave a model for inserting preventive maintenance into existing bus schedules in [15].

It can be seen that none of the above papers deal with multiple vehicle-specific activities. Moreover, the vehicle-specific activities that they study usually consider a single vehicle characteristic (e.g. vehicles will all have the same fuel type, or

require the same type of maintenance). In this paper, we introduce the multiple depot integrated vehicle scheduling and assignment problem with vehicle-specific tasks. This model acts as a general framework, where multiple activities with time or distance constraints can be included depending on the requirements of the specific problem. The model can provide feasible solutions with these constraints, while also taking capacity constraints into account. While all the above-mentioned papers solve the vehicle scheduling problems with a single specific vehicle activity, our goal was to provide a general model that can handle multiple activities simultaneously.

We will define the integrated vehicle scheduling and assignment problem with vehicle-specific activities (VSAP-VS) in the following way. Let the input of the problem be a set $T$ of timetabled trips, set $D$ of depots and set $V$ of vehicles. These concepts here are exactly the same as those introduced for the VSP at the beginning of this section; namely, trips have a departure and arrival time, a starting and ending location, and the vehicles can also be assigned to depots in the same way. Trips also determine the set of depots that are able to serve them by considering the features of the vehicles in the given depot.

In addition to the input given for the VSP, there are $n$ different vehicle-specific activities represented by set $R$. These activities are connected to vehicle characteristics rather than depots (e.g. refueling vehicles with natural gas and preventive maintenance of hybrid vehicles), and certain activities can only be carried out by given vehicles. Let set $R_j \subseteq R$ denote the possible tasks belonging to activity $j$. Similar to timetabled trips, task $r \in R_j$ has a starting and ending time, and departure and arrival locations (although these two are usually the same, as activities like parking or refueling are stationary). In order to model the capacities of the particular activities, they can only carried out in fixed, discrete time intervals instead of continuous availability. Each activity has its own set of rules and regulations, but these are usually connected to a set timespan or distance limit. For instance, a vehicle cannot travel more than a given distance without carrying out a refueling task, or it has to begin a parking task if it remains idle for more than a given amount of time. Compatibility between trips and activities can also be defined in a similar way to the VSP. Consider a pair of tasks $a, a' \in T \cup R$, which are compatible if:

- the same vehicle $v \in V$ is able to service both of them (the depot of $v$ is compatible with any trips in $\{a, a'\}$),
- they both satisfy the vehicle characteristics of $v$ (any vehicle-specific task in $\{a, a'\}$ can be carried out by $v$; e.g. we cannot assign battery recharging to a vehicle running on gas), and
- vehicle $v$ can service $a'$ after finishing $a$ with respect to the running time and distance between the arrival location of $a$ and the departure location of $a'$ (also considering the possible deadhead trip between $a$ and $a'$, where necessary).

The aim is to provide a feasible vehicle schedule that includes both timetabled trips and vehicle-specific activities, and tasks in any vehicle block are pairwise compatible. The cost of such a vehicle schedule is a linear combination of three different terms; namely, a one-time daily cost for each vehicle covering a block, a distance proportional cost for covering timetabled trips and deadheads, and costs of the vehicle-specific activities included in the blocks.

In the remainder of this paper, we will first introduce a mathematical model and solution method for the VSAP-VS, then demonstrate its efficiency by solving randomly generated test instances of different sizes and types.

# 3 Modeling and solution of the problem

In this section, we introduce our model for creating vehicles schedules that also take vehicle-specific activities into account. Then we present our solution method for it.

## 3.1 A set partitioning mathematical model

For each depot $d \in D$, let $B_d$ be the set of feasible $b$ blocks that start from $dt(d)$ and also return there. A block is a sequence of compatible trips and activities that can be executed by the same vehicle, and it satisfies all activity rules connected to this vehicle. Let

$$B = \bigcup_{d \in D} B_d$$

be the set of all such blocks. For each $b \in B_d$, let $y_b^d$ be the following binary variable

$$y_b^d = \begin{cases} 1, & \text{if } b \in B_d \text{ block is part of the solution} \\ 0, & \text{otherwise} \end{cases}$$

Furthermore, let $a_{e,b}^d$ be the following

$$a_{e,b}^d = \begin{cases} 1, & \text{if } b \in B_d \text{ block contains activity edge } e \\ 0, & \text{otherwise} \end{cases}$$

Let $T$ denote the set of trips for the problem, and $R$ give the set of tasks belonging to all vehicle-specific activities. Tasks that are connected to a single activity of a given vehicle characteristic are given by $R_i \subseteq R (1 \leq i \leq n)$, where $n$ is the number of all such activities. The capacity of a depot $d \in D$ is denoted by $k_d$, and the maximum number of vehicles that can simultaneously perform a vehicle-specific task $r \in R$ is given by $k_r$. Let $c_b$ be the cost associated with block $b \in B_d$. Then we can formulate the problem in the following way:

$$\text{minimize} \sum_{d \in D} \sum_{b \in B_d} c_b y_b^d, \tag{1}$$

s.t.

$$\sum_{d \in D} \sum_{b \in B_d, e=(dt(t),at(t)) \in E_d} a_{e,b}^d y_b^d = 1, \forall t \in T \tag{2}$$

$$\sum_{d \in D} \sum_{b \in B_d, e=(dt(r),at(r)) \in E_d} a_{e,b}^d y_b^d \leq k_r, \forall r \in R \tag{3}$$

$$\sum_{b \in B_d} y_b^d \leq k_d, \forall d \in D \tag{4}$$

$$y_b^d \in \{0,1\}, \forall d \in D, \forall b \in B_d \tag{5}$$

$$a_{e,b}^d \in \{0,1\}, \forall d \in D, \forall b \in B_d, \forall e \in E_d \tag{6}$$

Constraint (2) ensures that every trip is covered exactly once. Blocks simultaneously containing certain vehicle-specific activities are given by constraint (3), each task $r \in R$ having a maximum capacity $k_r$. Note that this constraint only ensures the vehicle limits on each task, as we suppose that every block is feasible, also meaning that they satisfy the distance and time constraints or any other rules connected to the activities. Managing this feasibility will be addressed later on. Finally, constraint (4) limits the capacities of each depot $d \in D$.

## 3.2 A column generation approach

Due to the huge number of possible blocks (resulting in a large amount of decision variables), the model cannot be solved directly by a MIP solver. Moreover, generating all blocks is also problematic, as the number of possible combinations is huge. Instead of giving all the possible blocks in the model, only the most important ones have to be generated to achieve a good quality solution.

Column generation [11, 20] is a classical method that is usually applied to such problems, relaxing the integer constraints of the variables. This relaxed problem is also known as the master problem. The usual steps taken during the solution process are the following:

1. Create an initial solution. The resulting schedule will provide the starting set of columns.
2. Solve the relaxed problem (master problem) on the actual set of columns, store the lower bound, and duals.
3. Solve a pricing problem in order to look for new columns that have a negative reduced cost.
4. Add the new columns to the master problem, and erase any old columns that are obsolete, and have a large cost.

5. Check the termination criteria. If none apply, go to step 2.

6. Create the final schedule based on the current columns of the problem.

Creating the final vehicle schedule in step 6 can be achieved by solving the resulting problem as an IP using a solver. A solver can also be used in step 2 for the solution of the master problem LP. The process can terminate in step 5 if a given iteration count is reached, or the solution has not improved significantly in the past few iterations. The most important part of the algorithm is the solution of the pricing problem in step 3.

## 3.3 Initial solution

Next, we present our heuristic for creating an initial solution for the column generation process. Its pseudo code can be seen in Algorithm 1.

The input of the algorithm is the set $T$ of trips and set $V$ of vehicles. The process iterates over the input trips in ascending order of their departure times, and assigns the current trip to an existing block with the cheapest cost. If there is no block where the trip can be inserted, then a new block is created for the trip. The vehicle chosen for this block is the one with the smallest cost that can execute the trip. After each trip assignment, the current block is checked to see whether its assigned vehicle has to undergo a task belonging to any of its vehicle-specific activities.

The function $checkActivity(A, b)$ checks block $b$ to see if it could execute any of the remaining trips without violating the rules of activity $A$. If any of the rules are violated, $b$ is sent to carry out the given activity, and is assigned the cheapest compatible task. If the activity was connected to a resource (e.g. distance or time), this is also replenished for the vehicle servicing the block.

As an example, we present a function for managing refueling activities in Algorithm 2. Vehicles are sent for refueling tasks if their remaining distance (denoted by *remDist*) does not allow them to head out for any trip, service it, and then head back to any location. For this, we count the distance of two deadheads as *maxDeadheadTime* (with the maximal possible distance), and the distance of the trip as *maxTripTime* (taking the maximal remaining trip distance into account). If refueling is needed, the vehicle is assigned to the next available possibility with the cheapest cost. After the refueling task is completed, the remaining distance that the vehicle can cover is again set to its maximum value.

When the initial solution is created, its blocks are used as the starting columns of the relaxed master problem.

## 3.4 Pricing problem

After solving the master problem, information about its duals is used to create new columns that can improve its current objective. Each such column corresponds to a legal vehicle block. These blocks are created with the use of a generation network.

**Algorithm 1** Initial vehicle schedules with activities.

**Funct** buildSchedule$(T, V)$

Let the set of blocks be $B := \{\}$
Order T by ascending trip departure times
**for** $(t \in T)$ **do**
  Let $f := b \in B$ with the cheapest insertion cost for $t$
  **if** $f = \emptyset$ **then**
    $f := v \in V$ that can serve $t$ with the smallest cost
    Assign $t$ to $f$
    $B := B \cup \{f\}$
  **else**
    Assign $t$ to $f$
  **end if**
  **for** all activities $R_i \subseteq R$ **do**
    checkAtivity$(R_i, f)$
  **end for**
**end for**
**return** $B$

**Funct** checkActivity$(A, b)$

$P :=$ all available tasks in A for $b$
$d :=$ resource (time/distance) needed for $b$ to serve any trip $t$
$v :=$ is an activity rule violated servicing any trip $t$?
**if** $d \geq remainingResource(b)$ **or** $v =$ **true then**
  $p :=$ cheapest compatible task from $P$
  Assign $p$ to $b$
  $remainingResource(b) := MAX$
**else if** v = true **then**
  $p :=$ cheapest compatible task from $P$
  Assign $p$ to $b$
**end if**

**Algorithm 2** Function for checking refueling activities.

**Funct** checkFuel$(b)$

$R :=$ all available refueling times for $b$
$d := 2 \cdot maxDeadheadTime + maxTripTime$
**if** $d \geq remDist(b)$ **then**
  $r :=$ cheapest compatible refueling possibility for $b$
  Assign $r$ to $b$
  $remTime(b) := MAX$
**end if**

This network provides the basis of the pricing problem, and it is used to build vehicle blocks with a negative cost that also satisfy all the above-mentioned vehicle-

specific activities. This basically means that after a vehicle has consumed enough of a given resource, the appropriate events for its vehicle-specific needs also have to be scheduled. This will be done by solving a resource-constrained shortest path problem.

We use a time-space generation network similar to the one presented by [24], and a separate network is created and solved for every pair of depot and activity type. Each network is given by the possible tasks that can be assigned to the vehicles, namely timetabled trips, deadhead trips and other vehicle-specific events. The nodes of the network correspond to the arrival and departure time of these tasks on their given time-lines. The edges of the network can either correspond to their respective tasks, or be waiting edges on the time-lines.

Formally, let $G = (N, E)$ represent a general duty generation network. Such a network is created for each combination of depot $d$ and activity $a$, resulting in networks $H_a^d = (N_a^d, E_a^d)$. The nodes in $N_a^d$ are the following: *depot_source, depot_sink, trip_start, trip_end, activity_start, activity_end*. Edges in $E_a^d$ connect these nodes; *trip_start* and *trip_end* nodes belonging to the same trip are connected by *trip edges*, *activity_start* and *activity_end* nodes belonging to the same activity are connected by *activity edges*. Any end node is connected to the start nodes of other tasks; *deadhead edges* connect those in different locations, while *waiting edges* run between nodes in the same location. The only exception to this is activities. Namely, the end node of an activity task is not connected to the start node of the same activity type, as there is no point in executing two similar vehicle-specific activities after each other. *Block_start edges* connect the *depot_source* node to every *trip_start* node, and every *trip_end* node is connected to the *depot_sink* node with *block_end edges*.

An example of such a network can be seen in Figure 1. Here, refueling is the vehicle-specific activity of the network, and it is performed in 20-minute tasks.

To ensure feasibility with respect to vehicle-specific events, so-called resources are also associated with each vehicle on the network. A single resource is allocated for each vehicle-specific activity, which calculates the time/distance (depending on the resource) covered by the vehicles with the help of a resource extension function. These will filter out blocks whose total consumed resources violate any of the vehicle-specific requirements. Tasks belonging to the activity replenish the appropriate resource capacity of the executing vehicle.

As described in [24], negative reduced cost vehicle blocks are generated on these networks using a dynamic programming approach presented in [12]. Blocks with the lowest negative reduced cost are added to the master problem from each generation network.

## 3.5 Creating the final schedule

After the column generation steps have terminated, the resulting master problem only gives us a solution for the LP-relaxed scheduling problem. To obtain a final integer solution, a second phase is usually executed. This can be done in several ways.

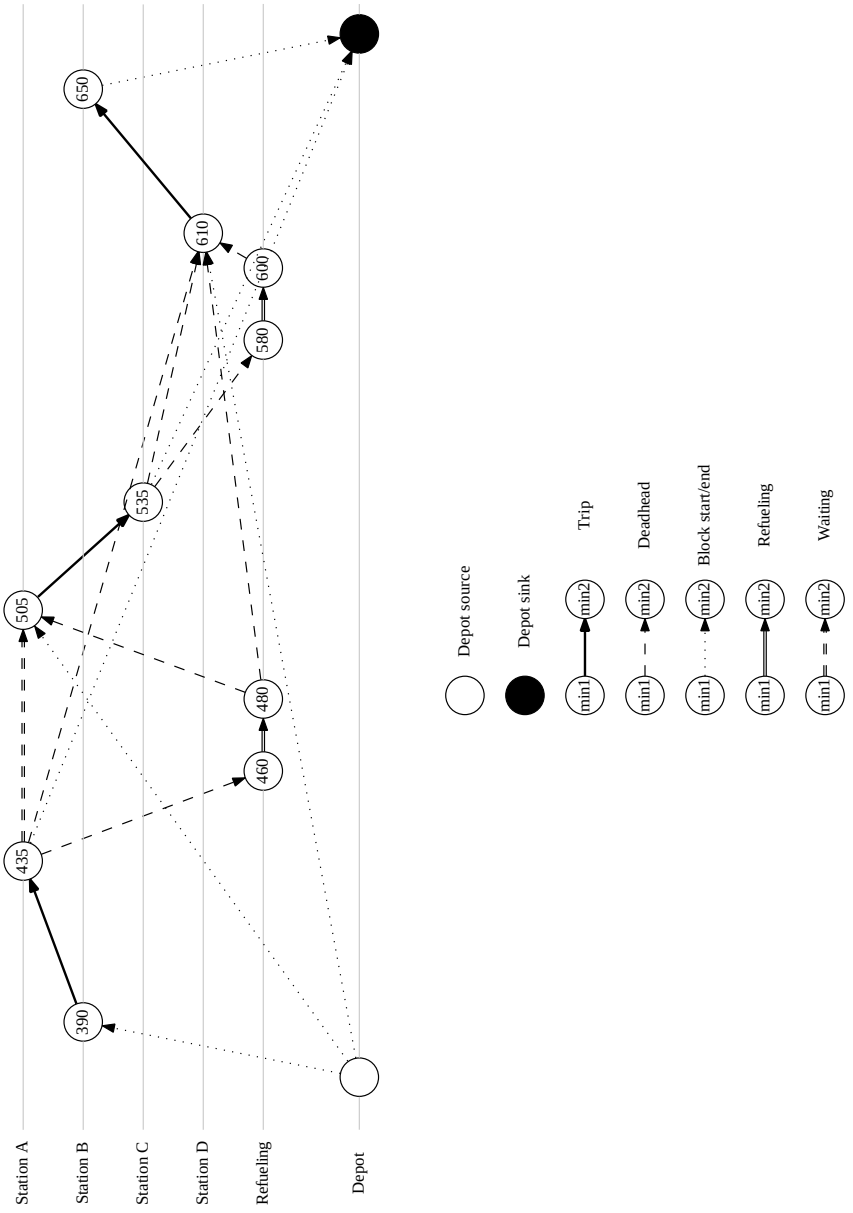



Figure 1: Example of a generation network

One approach is to use Lagrangian relaxation [16], as in the case of integrated vehicle and crew scheduling problems, where the linking constraints between vehicle and crew are an ideal candidate to be relaxed.

Another possibility is to embed the column generation process in a branch-and-bound framework that searches for the optimal integer solution [6, 21]. This method is also referred to as branch-and-price.

Integer solutions can also be obtained by the use of truncated column generation, which was used for the MDVSP by Pepin et al. [22]. After the column generation has terminated, this approach attempts to obtain an integer solution of the master problem by performing a rounding/variable fixing heuristic that fixes the values of fractional variables.

Our approach is similar to truncated column generation, but instead of using a heuristic approach to round the variables of our problem, we simply re-introduce the integrality constraint to our master problem, and solve it using an IP solver. A similar approach to this was also applied in [14]. To control the time of the solution process, we also set a time limit for the solver.

# 4 Test results

We tested our method on random input generated based on [9]. Both single and multiple-depot test cases were created in different sizes; namely 50, 250, 500, 1000, 1500 and 2000 trips. While the model we presented can be used as a general framework for handling different vehicle-specific activities, we chose refueling to showcase our test instances, as this is the most widely studied vehicle activity. However, while papers in the literature usually deal with only a single fuel type for a problem, our test instances use two different fuel types. Moreover, we allow vehicles belonging to the same depot to have different types of fuel, which is rarely considered in other studies.

A total of 24 test cases were solved both for the single- and multiple depot problems; we generated 4 instances for each problem size. The ILOG CPLEX solver was used during the solution process, with a limit set on its running time: i.e. the limit on the column generation phase was 7.5 hours, while the IP phase ran for 3 hours in the single depot case, and 5 hours in the multi-depot case. This gave a total maximum running time of 10.5 hours (37800 seconds) for single depot problems, and 12.5 hours (45000 seconds) for multi-depot problems. These limits have to be included if we consider the practical aspects of such a solution method, as planners should have an estimate on the required running time if they want to have a feasible solution.

The following types of data are presented for our results: the number of vehicle blocks given by the initial heuristic (initial blocks), the final number of blocks given by the resulting IP (IP blocks), the final optimality gap (%) given by CPLEX, and the solution time of the instance.

Table 1 shows results obtained for the single depot test instances. It can be seen that good quality solutions are achieved for all 24 problems, the optimality gap is

Table 1: Single depot test instances.

| Instance | Trips | Initial blocks | IP blocks | MIP gap (%) | Time (s) |
|---|---|---|---|---|---|
| input_01 | 50 | 16 | 16 | 0.00% | 95 |
| input_02 | | 16 | 10 | 0.00% | 80 |
| input_03 | | 16 | 11 | 0.00% | 86 |
| input_04 | | 16 | 16 | 0.00% | 85 |
| input_05 | 250 | 58 | 71 | 0.01% | 258 |
| input_06 | | 56 | 56 | 0.00% | 335 |
| input_07 | | 58 | 58 | 0.01% | 329 |
| input_08 | | 56 | 69 | 0.00% | 257 |
| input_09 | 500 | 99 | 131 | 0.08% | 4938 |
| input_10 | | 107 | 104 | 0.43% | 5473 |
| input_11 | | 103 | 107 | 0.49% | 5527 |
| input_12 | | 96 | 137 | 0.18% | 5014 |
| input_13 | 1000 | 184 | 290 | 1.51% | 11228 |
| input_14 | | 183 | 231 | 2.16% | 19808 |
| input_15 | | 187 | 219 | 2.03% | 19361 |
| input_16 | | 185 | 288 | 1.54% | 11073 |
| input_17 | 1500 | 260 | 611 | 0.01% | 21028 |
| input_18 | | 268 | 529 | 3.56% | 21629 |
| input_19 | | 264 | 612 | 0.01% | 18488 |
| input_20 | | 262 | 595 | 2.05% | 21612 |
| input_21 | 2000 | 374 | 731 | 2.70% | 21643 |
| input_22 | | 364 | 663 | 0.76% | 21615 |
| input_23 | | 358 | 383 | 0.00% | 18105 |
| input_24 | | 358 | 593 | 0.01% | 21007 |

generally under 1% (with only a single instance being above 3%, and six instances lying between 1% and 3%). The maximum runtime limit was exceeded only by the 1500 and 2000 trip instances. One notable feature of all single depot results is that the IP solution of the problem uses significantly more buses than the initial heuristic. The reason for this can be found by examining the different cost factors of the input. Most of the vehicles generated for the input instances ended up having a relatively low daily cost, and a more significant distance proportional cost factor. Because of this, the IP solution attempted to minimize the distance traveled by its vehicles (which meant trying to schedule as few deadhead trips as possible). The initial heuristic is essentially a greedy assignment of trips to vehicles, which does not take the distance traveled into account, and only introduces new vehicles to the schedules if it really has to.

Overall, the solutions we obtained for a single depot and two fuel types look

favorable, even for larger instances. The maximum problem size presented by the test results for similar problems in other papers in the literature rarely exceed 1000 trips, while we obtained good quality solutions for several instances with 1500 and 2000 trips as well, taking into account vehicles with two different refueling constraints.

Table 2: Multiple depot test instances.

| Instance | Trips | Initial blocks | IP blocks | MIP gap (%) | Time (s) |
|---|---|---|---|---|---|
| input_1 | 50 | 26 | 9 | 0.00% | 10 |
| input_2 | | 25 | 8 | 0.00% | 16 |
| input_3 | | 10 | 9 | 0.01% | 26 |
| input_4 | | 9 | 9 | 0.00% | 13 |
| input_5 | 250 | 51 | 32 | 8.06% | 19028 |
| input_6 | | 46 | 34 | 6.42% | 19301 |
| input_7 | | 47 | 35 | 7.30% | 19253 |
| input_8 | | 45 | 35 | 6.49% | 18998 |
| input_9 | 500 | 84 | 64 | 11.43% | 22801 |
| input_10 | | 100 | 63 | 10.84% | 23184 |
| input_11 | | 81 | 57 | 12.73% | 24836 |
| input_12 | | 86 | 58 | 12.48% | 24452 |
| input_13 | 1000 | 181 | 135 | 15.29% | 45067 |
| input_14 | | 164 | 122 | 23.59% | 45045 |
| input_15 | | 177 | 124 | 21.65% | 45087 |
| input_16 | | 182 | 124 | 23.48% | 45064 |
| input_17 | 1500 | 281 | 216 | 25.11% | 45081 |
| input_18 | | 270 | 217 | 26.64% | 45033 |
| input_19 | | 291 | 221 | 27.02% | 45024 |
| input_20 | | 284 | 222 | 27.59% | 45112 |
| input_21 | 2000 | 366 | 320 | 26.35% | 45113 |
| input_22 | | 376 | 329 | 26.10% | 45160 |
| input_23 | | 378 | 336 | 26.19% | 45119 |
| input_24 | | 335 | 313 | 27.60% | 45061 |

Table 2 gives the test results for the multi-depot test cases. Here we used vehicles belonging to two different depots, and we also considered two fuel types (both depots had a mix of vehicles with both fuel types). This results in a more complicated problem (multiple depots, two fuel types, instances with a large number of trips) than those that are usually examined in the literature.

The results for small, 50-trip instances look promising, as we also found near optimal solutions in a short time. However, these were the only cases where both the column generation phase and the IP phase terminated well before its time limit.

Even for the 250 trip instances, the IP solution process was aborted prematurely as it ran out of time. For the 250 and 500 trip instances, the column generation process terminated with no more columns to generate, but it also ran out of time for the larger input.

The effect of reaching the time limit can clearly be seen in the results. While the average gap given by the solver was 9.47% for the 250- and 500-trip instances, where the column generation finished without any problems, the three remaining instance sets (with 1000, 1500 and 2000 trips), where both the column generation and IP solution phases were aborted because they exceeded the time limit, had an average gap of 24.72%. While these results might not seem particularly promising, the quality of the solution obtained only depends on the time allocated for the two phases. Based on the single depot, and small multi-depot results, the model will provide better quality solutions, but the time limit has to be increased significantly. However, this would invalidate the very reason that we introduced the time limit in the first place; namely, to ensure that the results are useful in practice, which means that they have to be produced in a reasonable time.

## 5 Conclusions and future work

In this paper, we presented a general framework for the integrated vehicle scheduling and assignment that also takes into account tasks for vehicle-specific activities. This framework provides a daily vehicle schedule that also includes the special needs of the vehicles executing it; activities such as refueling and parking are considered in the resulting vehicle blocks. We presented a set partitioning-based mathematical model for the problem, where most vehicle-specific activities can easily be integrated based on the desired constraints. This model is then solved using a column generation approach. We demonstrated the efficiency of the proposed model on randomly generated test instances, using refueling to showcase vehicle-specific activities.

Instances with one and two depots were both generated, and all of them had two fuel types with different constraints (also allowing the possibility that the same depot can contain vehicles with different fuel types). A time limit was set on both phases of the solution process to guarantee that a feasible result could be achieved in a reasonable time. Test runs for a single depot and two fuel types resulted in good quality solutions, as did the smaller-sized multi-depot ones. The results for the larger multi-depot instances had a bigger optimality gap, but this was due to one or both of the phases terminating because of the time limit set. Based on the results of the other test cases, these gaps should also be smaller if more time is provided for finding the solution.

While the results are promising, there is still room for improvement of the process. Implementing a proper branch-and-price framework could result in a better quality solution. Because of the limited running time, a truncated column generation approach with a rounding heuristic as the second phase should also be considered. Another way of reducing the solution time might be the parallelization

of the column generation process. Implementing these approaches and comparing their results should be the next phase of this work.

The model we created is a general, flexible framework that can handle multiple vehicle activities. Although we presented problems with two different fuel types as our test cases, we did not generate instances with two or more completely different activity types. Our future research work will also include experimenting with different vehicle constraints connected to these activities.

## Acknowledgements

József Békési was supported by the EU-funded Hungarian grant EFOP-3.6.2-16-2017-00015.

Balázs Dávid was supported by the project "Integrated program for training new generation of scientists in the fields of computer science", no EFOP-3.6.3-VEKOP-16-2017-0002. The project has been supported by the European Union and co-funded by the European Social Fund.

Miklós Krész was supported by National Research, Development and Innovation Office - NKFIH Fund No. SNN-117879.

## References

[1] Adler, Jonathan D. *Routing and scheduling of electric and alternative-fuel vehicles*. PhD thesis, Arizona State University, 2014.

[2] Adler, Jonathan D and Mirchandani, Pitu B. The vehicle scheduling problem for fleets with alternative-fuel vehicles. *Transportation Science*, 51(2):441–456, 2016.

[3] Bertossi, Alan A, Carraresi, Paolo, and Gallo, Giorgio. On some matching problems arising in vehicle scheduling models. *Networks*, 17(1):271–281, 1987.

[4] Bodin, Lawerence and Golden, Bruce. Classification in vehicle routing and scheduling. *Networks*, 11(1):97–108, 1981.

[5] Bodin, Lawerence, Golden, Bruce, Assad, Arjang, and Ball, Mark. Routing and scheduling of vehicles and crews: The state of the art. *Computers and Operations Research*, 10(1):63–212, 1983.

[6] Borndörfer, Ralf, Löbel, Andreas, and Weider, Steffen. A bundle method for integrated multi-depot vehicle and duty scheduling in public transit. *Lecture notes in economics and mathematical systems*, 600:3, 2008.

[7] Borndörfer, Ralf, Reuther, Markus, Schlechte, Thomas, Waas, Kerstin, and Weider, Steffen. Integrated optimization of rolling stock rotations for intercity railways. *Transportation Science*, 50(3):863–877, 2015.

[8] Bunte, Stefan and Kliewer, Natalia. An overview on vehicle scheduling models. *Journal of Public Transport*, 1(4):299–317, 2009.

[9] Dávid, Balázs and Krész, Miklós. Application oriented variable fixing methods for the multiple depot vehicle scheduling problem. *Acta Cybernetica*, 21(1):53–73, 2013.

[10] Dávid, Balázs and Krész, Miklós. The dynamic vehicle rescheduling problem. *Central European Journal of Operations Research*, 25(4):809–830, 2017.

[11] Desaulniers, Guy, Desrosiers, Jacques, and Solomon, Marius M. *Column generation*, volume 5. Springer Science & Business Media, 2006.

[12] Desrosiers, Jacques, Dumas, Yvan, Solomon, Marius M, and Soumis, François. Time constrained routing and scheduling. *Handbooks in operations research and management science*, 8:35–139, 1995.

[13] Freling, Richard and Pinto Paixão, José M. *Vehicle Scheduling with Time Constraint*, pages 130–144. Springer Berlin Heidelberg, 1995.

[14] Gintner, Vitali, Kliewer, Natalia, and Suhl, Leena. *A Crew Scheduling Approach for Public Transit Enhanced with Aspects from Vehicle Scheduling*, pages 25–42. Springer Berlin Heidelberg, 2008.

[15] Haghani, Ali and Shafahi, Yousef. Bus maintenance systems and maintenance scheduling: model formulations and solutions. *Transportation Research Part A: Policy and Practice*, 36(5):453–482, 2002.

[16] Huisman, Dennis. *Integrated and dynamic vehicle and crew scheduling*. PhD thesis, Erasmus University Rotterdam, 2004.

[17] Kuby, Michael and Lim, Seow. The flow-refueling location problem for alternative-fuel vehicles. *Socio-Economic Planning Sciences*, 39(2):125–145, 2005.

[18] Li, Jing-Quan. Transit bus scheduling with limited energy. *Transportation Science*, 48(4):521–539, 2013.

[19] Li, Jing-Quan. Battery-electric transit bus developments and operations: A review. *International Journal of Sustainable Transportation*, 10(3):157–169, 2016.

[20] Lübbecke, Marco E and Desrosiers, Jacques. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[21] Mesquita, Marta, Paias, Ana, and Respício, Ana. Branching approaches for integrated vehicle and crew scheduling. *Public Transport*, 1(1):21–37, 2009.

[22] Pepin, Ann-Sophie, Desaulniers, Guy, Hertz, Alain, and Huisman, Dennis. A comparison of five heuristics for the multiple depot vehicle scheduling problem. *Journal of Scheduling*, 12(1):17–30, 2009.

[23] Reuer, J, Kliewer, N, and Wolbeck, L. The electric vehicle scheduling problem: A study on time-space network based and heuristic solution approaches. In *Proceedings of the 13th Conference on Advanced Systems in Public Transport (CASPT), Rotterdam*, 2015.

[24] Steinzen, Ingmar, Gintner, Vitali, Suhl, Leena, and Kliewer, Natalia. A time-space network approach for the integrated vehicle-and crew-scheduling problem with multiple depots. *Transportation Science*, 44(3):367–382, 2010.

[25] van Kooten Niekerk, M. E., van den Akker, J. M., and Hoogeveen, J. A. Scheduling electric vehicles. *Public Transport*, 9(1):155–176, 2017.

*Received 7th May 2018*

Acta Cybernetica **23** (2018) 801–813.

# $\ell_1$ Regularization of Word Embeddings for Multi-Word Expression Identification

Gábor Berend[a]

**Abstract**

In this paper we compare the effects of applying various state-of-the-art word representation strategies in the task of multi-word expression (MWE) identification. In particular, we analyze the strengths and weaknesses of the usage of $\ell_1$-regularized sparse word embeddings for identifying MWEs. Our earlier study demonstrated the effectiveness of regularized word embeddings in other sequence labeling tasks, i.e. part-of-speech tagging and named entity recognition, but it has not yet been rigorously evaluated for the identification of MWEs yet.

**Keywords:** sparse coding, multi-word expressions, word embeddings

## 1 Introduction

Multi-word expressions (MWEs) are semantically coherent linguistic constructions including whitespace characters like "*paternal leave*" and "*shut off*" [11, 22]. The identification and proper treatment of such expressions is an important and challenging task which can improve the performance of various natural language processing (NLP) applications such as the extraction of opinionated expressions [2] and machine translation [6].

Continuous word embeddings have become prevalent in a variety of NLP tools due to their intriguing property of being able to capture both semantic and syntactic properties of word forms [15]. Such dense word representations have been successfully applied in many NLP analyzers such as syntactic parsers and part-of-speech taggers [8, 19, 20].

Instead of the typical approach of regarding the dense vectorial representations of words as the discriminative features, here we investigate the utilization of $\ell_1$ regularized sparse word embeddings, which has been shown to provide substantial gains in the tasks of part-of-speech (POS) tagging and named entity recognition (NER) [3] over multiple languages. Besides utilizing regularized word embeddings we do not rely on any other (linguistic) resources in order to keep the proposed approach easily applicable to new languages.

[a]Department of Informatics, University of Szeged, E-mail: berendg@inf.u-szeged.hu

DOI: 10.14232/actacyb.23.3.2018.5

## 2 Previous work

In [29], the authors use a sequence labeling framework for the detection of a special kind of MWE, namely light verb constructions. In [23], a wider range of MWEs are studied by applying a standard chunking representation and proposing a feature-rich discriminative sequence tagging algorithm for the proposed problem. The feature-rich representation of typical approaches often assume the existence of additional linguistic resources, such as gazetteers containing highly indicative words for certain kinds of MWEs, part-of-speech taggers and even syntactic parsers [27]. While the use of such external resources is legitimate from a linguistic perspective, it makes these approaches less robust for utilizing them in languages where such resources do not exist.

Word embeddings, however, are capable of representing the syntactic and/or semantic nature of word forms and can be trained in an unsupervised manner [15]. For this reason, sequence labeling models which rely on word embeddings can implicitly incorporate syntactic/semantic knowledge without an explicit reliance of NLP parsers. Word embeddings have became commonly used in many MWE-related tasks due to their intriguing properties. For instance, word embeddings are user in order to improve the quality of the translation of phrasal verbs in [7].

The authors of [20] contrast the effects of utilizing differently trained dense word embeddings and Brown clustering for the application of classical uni-and bigram-based models in MWE identification besides part-of-speech tagging, syntactic chunking and named entity recognition. They found that models which had access to word embeddings had a consistent advantage over models which classified tokens based on unigram features. At the same time they report that there was no word embedding approach that would have a clear advantage over the others for all the sequence labeling tasks and that one can perform competitively with models that rely on continuous word embeddings for certain sequence labeling tasks by relying on Brown cluster identifiers of word forms.

Our earlier work has demonstrated that substantial improvements can be gained in the tasks of part-of-speech tagging and named entity recognition if the discriminative features that are used by the sequence classifiers are derived from the $\ell_1$ regularized variants of dense word representations instead of the dense vectorial representation of word forms [3]. In this study, we investigate and rigorously compare the applicability of this approach for the task of MWE identification.

Unsupervised word clusters (e.g. in the form of Brown clustering [5]) have also been frequently employed for representing words in various sequence labeling tasks for NER [21, 9], chunking [26], POS tagging [24] and MWE identification [23].

MWEs are in the focus of multiple other research efforts. The approach presented in [4] is among the alternatives for acquiring multiword lexicons in an unsupervised manner using n-gram lattices.

# 3 Experimental settings

The experimental setting in this study extends that of [3], where we showed that sequence models relying on features derived from the $\ell_1$ regularized versions of dense word embeddings perform competitively or even better than classical models for part-of-speech tagging and named entity recognition. We released the code base used in our experiments at `https://github.com/begab/tacl_sparse_coding`.

## 3.1 Applying $\ell_1$ regularized word embeddings

The approach described in [3] relies on continuous word embeddings, such as word2vec [15] and Glove [18]. Word embeddings map the symbolic elements of the vocabulary of some language to $m$-dimensional real-valued vectors ($\mathbf{x} \in \mathbf{R}^m$) such that syntactically and/or semantically similar word forms get assigned vectors which point in similar directions. For a vocabulary consisting of $n$ distinct word forms, these word embeddings can be stacked to form a $X \in \mathbf{R}^{m \times n}$ matrix. Such word embeddings can be constructed with a variety of open-source tools[1][2] and require no resources other than raw, unannotated text corpora for which reason their usage has become ubiquitous in many NLP applications.

The $\ell_1$ regularization of word embeddings takes place using dictionary learning [14], which decomposes the original embedding matrix $X$ by solving the following optimization problem

$$\min_{D \in \mathcal{C}, \alpha} \|X - D\alpha\|_F^2 + \lambda \|\alpha\|_1, \tag{1}$$

in which $\mathcal{C}$ is the convex set of matrices of column vectors having an $\ell_2$ norm of at most one, matrix $D \in \mathbf{R}^{m \times k}$ acts as the shared dictionary across the word embedding signals, and the columns of the sparse matrix $\alpha \in \mathbf{R}^{k \times n}$ contain the coefficients for the linear combinations of each of the $n$ observed signals.

Dictionary learning has two parameters, namely $k$, which is the number of basis vectors to be included in the dictionary matrix $D$, and the regularization coefficient $\lambda$, which implicitly controls the amount of non-zero coefficients in $\alpha$; that is, the amount of basis vectors utilized in the reconstruction of input word embeddings. Assuming that the vectorial representation of some word form $x$ is located in the $i^{th}$ column of the embedding matrix $X$, sparse discriminative features are derived from those positions of the $i^{th}$ column of $\alpha$ that contain non-zero coefficients. In the remainder of the paper, we shall refer to sequence classifiers which assign discriminative features to word forms this way as sparse models.

In contrast to sparse models, scalars comprising the original dense vectors assigned to word forms can act as discriminative features as well. This means that each token is described by $m$ scalars, whereas in the sparse scenario tokens are described by a fraction of indicator variables depending on the regularization parameter chosen. We shall refer to sequence classifiers that treat word forms this way as dense models.

[1] `https://code.google.com/archive/p/word2vec/`
[2] `http://nlp.stanford.edu/projects/glove/`

### 3.2 The dataset

The dataset that we conducted our experiments on is the WIKI50 corpus [28]. WIKI50 is a collection of 50 Wikipedia articles in which all the occurrences of 4+6 different kinds of multi-word units have been annotated manually. Proper nouns often consist of multiple tokens, which is why the dataset contains annotations for the 4 standard named entity (NE) categories, i.e. *Organization*, *Person*, *Location* and *Miscellaneous*. The dataset also distinguishes the following MWEs (with examples in parenthesis): *Noun Compounds* ("public transportation"), *Adjectival Compounds* ("monkey styled"), *Verb-Particle Constructions* ("went on"), *Light-Verb Constructions* ("opens fire"), *Idioms* ("caught the eye of") and *Other* ("alter ego"). The dataset consists of 114,284 tokens and 4366 sentences originating from 50 Wikipedia articles.

When reporting detailed results for the individual MWE classes we focus on 8 different types of MWEs, as opposed to the 10 total classes distinguished in the WIKI50 corpus. This is due to the fact that we do not report results for the MWE categories *Idiom* and *Other* due to their highly infrequent nature. The above–mentioned categories have 19 and 21 occurrences over the entire WIKI50 dataset, respectively, meaning that more than half of the Wikipedia articles do not contain a single instance of these categories. The overall classification metrics that we report, however, do incorporate results on these two categories as well.

## 4 Experimental results

We use the CRFsuite [16] package to train first-order conditional random fields (CRF) [12] models as sequence classifiers. Unless otherwise stated, words at a certain position within a sequence are described by the (sparse or dense) features representing the given word and also those of its immediate neighbors. Features also incorporate relative token positions (whether a certain feature comes from the previous, actual or the successive token) that were taken into consideration. This means that for the dense model each token position is described by a vector in $\mathbf{R}^{3m}$.

The performance of the models we experiment with is evaluated using 50-fold cross-validation. Here we train 50 models, that is for making predictions for a Wikipedia article taken from the WIKI50 corpus we train one model based on the labeled token sequences of all the remaining 49 Wikipedia articles from the dataset. This way when making predictions about the tokens of a particular Wikipedia article, we can ensure that none of the sentences from the same Wikipedia article is used during the parameter estimation of the model making predictions for the given document.

For evaluation purposes, we used the same script that was released as part of the 2002 CoNLL shared task on named entity recognition [25]. Even though the script was released for a shared task on NER, it seamlessly adapts for any set of class labels. It provides precision, recall and F-score metrics for the individual MWE types and also for the entire sequence labeling task.

## 4.1 Comparing sparse and dense embedding-based models

In this section, we investigate the effects of deriving features from dense versus sparse word embeddings for the sequence classification model. We experimented with four popular continuous word embeddings, i.e. glove [18], polyglot [1], skip-gram (sg) and continuous bag-of-words (cbow) [15]. As for the polyglot embeddings we use the publicly available [3] 64-dimensional pre-trained embeddings, which are trained over an English Wikipedia dump also made accessible by the authors of [1]. In order to be able to objectively assess the quality of word embedding techniques it is vital that the embeddings should be trained under as similar circumstances as possible. For this reason we trained our own sg, cbow and glove embedding over the same corpus that is used for training polyglot embeddings.

When deriving sparse word representations like that described in Section 3.1, we set $k$, the number of basis vectors in the dictionary matrix $D$, to 1000 and choose the value for $\lambda$ from $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. Depending on the value for $\lambda$, we found 0.5% to 5% of the coefficients in $\alpha$ to be non-zero, which means that the average number of features per word forms is between 5 (for $\lambda = 0.5$) and 50 (for $\lambda = 0.1$).

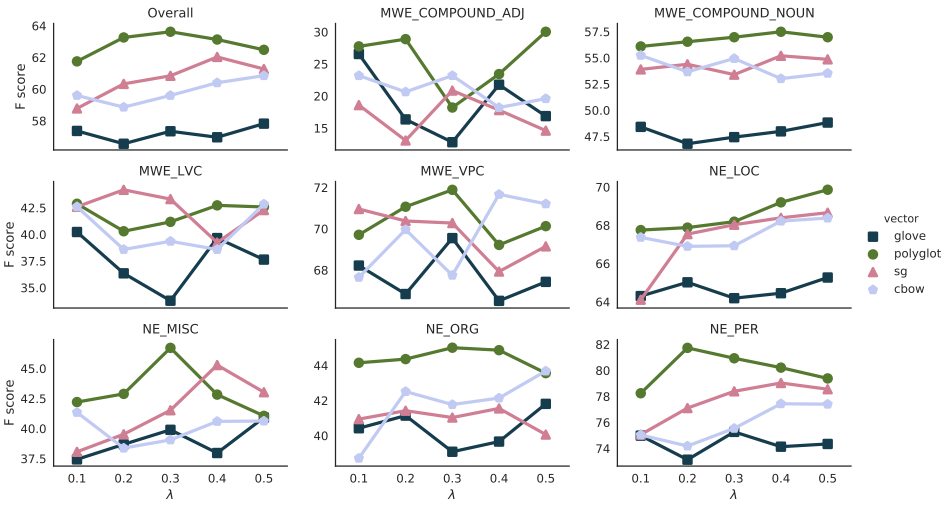



Figure 1: Overall results and a per-multi-word unit category breakdown of the F-scores as a function of the regularization parameter $\lambda$ and the pre-trained word embedding algorithms.

Figure 1 contains results for the overall classification performance and its breakdown according to the different MWE classes (excluding *Idioms* and *Other* class, as discussed in Section 3.2). This table tells us that the overall performance peaks for polyglot word embeddings with a regularization coefficient of 0.3. This choice of the regularization parameter provides not only the best overall F-scores, but it

[3] `https://sites.google.com/site/rmyeid/projects/polyglot`

produces the best performance for multiple individual MWE types. The class of compound adjectives behaves in the least predictable way when altering the regularization coefficient $\lambda$. This is due to the fact that this MWE type is one of the least frequent classes, for which reason the misclassification of a few instances can have a dramatic effect overall. Increasing $\lambda$ beyond a certain value (typically 0.3) has a detrimental effect on the performance for nearly all of the MWE types. The location NE category is a notable exception to this, as the identification performance for this category does not seem to degrade even for the highest level of regularization employed. Based on the entire contents of Figure 1, the regularization coefficient was set to 0.3.

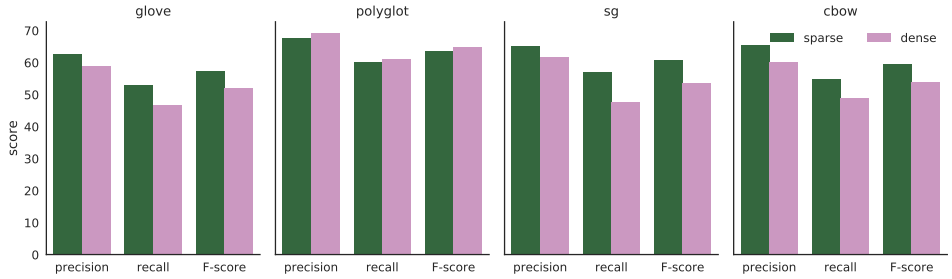



Figure 2: Overall results of models that utilize dense and sparse word embeddings-based features.

Figure 2 indicates that for most of the embedding types the features derived from sparse embeddings have a substantially better overall performance. The only exception is for polyglot embeddings where the sparse versions do not provide better results compared to the sequence classifier deriving features based on dense vectors. Figure 2 also indicates that polyglot embeddings obtained the best results for the task of MWE identification, hence comparative results in the remainder of the paper will be based on them.

## 4.2 Alternative models

In order to compare our word embedding-based results, we provide a variety of alternative approaches that will be presented and assessed next.

**Feature-rich representation** As an alternative to word embedding-based models, we evaluate a sequence classification model using a standard inventory of surface form features derived from the word identities themselves. The pool of feature templates is inspired by those made publicly available as part of CRFsuite [16]. We will use all the feature templates[4] included in the CRFSuite library, which derive features from word forms themselves but we do not include those features which are based on characters and character sequences comprising a word form. We omitted

[4] `https://github.com/chokkan/crfsuite/blob/master/example/pos.py`

character-based features as our primarily intention here is to compare the effects of word forms-derived features on sequence classification. The set of feature templates is listed in Table 1. Due to the high number of features induced by the templates, we shall refer to the models relying on them as feature rich (FR) models.

| Feature template | |
|---|---|
| $w_{t+j}$ | $-2 \leq j \leq 2$ |
| $w_t \oplus w_{t+j}$ | $1 \leq j \leq 9$ |
| $w_t \oplus w_{t-j}$ | $1 \leq j \leq 9$ |
| $\oplus_{i=t+j}^{t+j+1} w_i$ | $-2 \leq j \leq 1$ |
| $\oplus_{i=t+j}^{t+j+2} w_i$ | $-2 \leq j \leq 0$ |
| $\oplus_{i=t+j-1}^{t+j+2} w_i$ | $-1 \leq j \leq 0$ |
| $\oplus_{i=t-2}^{t+2} w_i$ | |

Table 1: Feature templates applied by our feature-rich baselines for some target word $w_t$ at position $t$ within a sequence. $\oplus$ is a binary operator forming a feature from words and their relative positions within the sequence by concatenating them.

**Brown clustering** Brown clustering [5] is widely used to provide useful word representation in many NLP sequence labeling tasks [17, 9, 21, 24]. For this reason we also train a sequence classifier for identifying MWEs which represents word forms based on their Brown cluster identifier.

In our experiments, we used the implementation of [13] to perform Brown clustering[5]. The same Wikipedia articles which serve as input for learning word embeddings are employed for determining 1024 Brown clusters over the vocabulary. The word features that we derive from the Brown cluster identifiers of word forms are the $\{4, 6, 10, 20\}$-long prefixes of Brown cluster identifiers of the word forms.

**Long-short term memory (LSTM) networks** LSTMs [10] are an extension of recurrent neural networks (RNN), which provide a remedy for the vanishing/exploding gradient problem during backpropagation in RNNs via gating mechanisms. LSTMs are regarded as the state-of-the-art models for many sequence labeling tasks in NLP.

The authors of [19] released their bidirectional LSTM implementation for part-of-speech tagging[6]. We adapted their implementation for training bi-LSTM sequence classifiers to identify MWEs. We made two modifications to their default settings, i.e. we used word embedding features only (whereas [19] defines character-level embeddings as well) and we trained the model for 15 epochs (instead of 30). The reason why we did not employ character-level embeddings in our model was that we wanted to compare the effects of various word representations alone and not to conflate it with the joint usage of additional features. What is more, the

[5] `https://github.com/percyliang/brown-cluster`
[6] `https://github.com/bplank/bilstm-aux`

use of character-level embeddings would make the training procedure substantially slower (especially that we performed 50-fold cross–validation).

We initialize the word embeddings of the bi-LSTM model with polyglot embeddings; however, they were treated as the parameters of the model, meaning that they were updated during training. The pre-initialization step of the word embedding parameters of the model is essential for good performance as the Wiki50 corpus is too small to learn reliable word embeddings based on it alone from a randomly initialized state. We observed that evaluation metrics of the bi-LSTM model degrade substantially if pre-initialization is not applied.

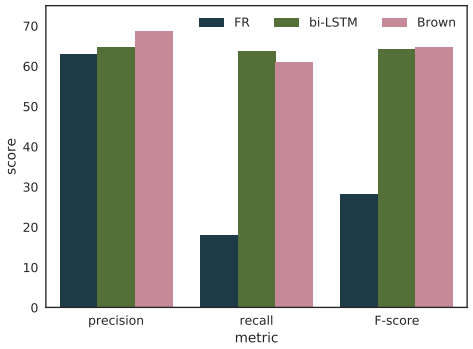



Figure 3: Overall results of the alternative models.

Figure 3 shows the overall results for the alternative models introduced previously. We observe that the feature–rich model performs the poorest mostly due to its low recall score. Another observation is that even though bi-LSTMs are considered as state-of-the-art approaches for sequence classification, it slightly underperforms the Brown clustering-based model, i.e. the bi-LSTM has an overall F-score of 64.22 as opposed to 64.90.

It should be mentioned that we managed to improve the scores of the bi-LSTM model by incorporating not only word embeddings, but character-level embeddings as well. Extending the model this way resulted in an overall F-score of 66.48 at the expense of a much longer training time. Furthermore, when we investigated the MWE-type specific changes in the scores, we realized that the overall improvement was due to improvements just for the named entity categories, whereas its ability to detect other types of non-NE MWEs either remained the same or even degraded slightly.

## 4.3 Detailed comparative results of different models

In order to gain a better insight into the performance of various models using conceptually different feature representations, we shall provide an MWE type-specific breakdown of the overall results. Table 2 provides an overview of the different models we investigated in our experiments. Inspecting Table 2, we see that precision

values tend to be higher than the recall scores with all the approaches and the bi-LSTM model seems to be the most balanced with respect to the gap between precision and recall scores.

| method | Precision | Recall | F-score |
|---|---|---|---|
| polyglot sparse ($\lambda = 0.3$) | 67.65 | 60.03 | 63.61 |
| bi-LSTM | 64.81 | 63.64 | 64.22 |
| Brown | 68.79 | 60.90 | 64.60 |
| polyglot dense | 69.24 | 61.07 | 64.90 |

Table 2: Comparison of the overall performance of conceptually different models.

Figure 4 includes the MWE-type specific breakdown of the individual models, which confirm that overall precision values tend to be higher compared to the recall scores. The only notable exception is the performance of the bi-LSTM model on the compound nouns, for which the precision scores are markedly lower compared to recall. This is due to the fact that the bi-LSTM model has a better ability to predict that category.

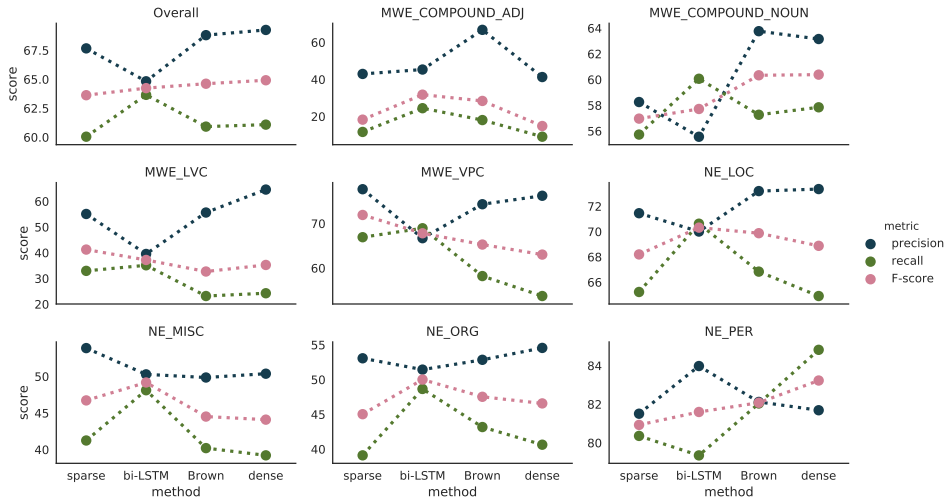



Figure 4: A per-MWE type comparison of the best performing models.

Figure 4 also elucidates the balanced nature of the bi-LSTM model in terms of the difference in precision and recall scores. The only exception to this balanced performance is the person NE type, for which it produces the highest precision–recall gap as the bi-LSTM model is less capable of predicting that particular category.

Looking at Figure 4 further, we can identify certain MWE categories for which certain approaches perform much better than others. The bi-LSTM has the best performance for named entity types apart from the person (NE_PER) category, the

Brown and dense models perform better than the other approaches for the compound noun category, whereas the sparse model achieves the best scores for the identification of light verb constructions (LVC) and verb-particle constructions (VPC).

## 5 Conclusions

In this paper, we investigated the applicability of sparse coding derived word features for the extraction of MWEs. Our experimental results demonstrate that the integration of sparse word features into sequence classifiers gives a performance competitive with state-of-the-art models, including bi-directional LSTMs. We should mention that the models applied here did not rely on POS taggers, syntactic parsers or gazetteers, implying that they can be conveniently adapted for the identification of MWEs in multiple languages without the need for any additional linguistic resources. Lastly, we found that sparse word representations seem to be the most suitable for the identification of verb-particle constructions and light verb constructions.

## Acknowledgement

We gratefully acknowledge the support of the NVIDIA Corporation with the donation of the Titan X Pascal GPU used in this research.

## References

[1] Al-Rfou, Rami, Perozzi, Bryan, and Skiena, Steven. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192. Association for Computational Linguistics, 2013.

[2] Berend, Gábor. Opinion expression mining by exploiting keyphrase extraction. In *Fifth International Joint Conference on Natural Language Processing, IJCNLP 2011, Chiang Mai, Thailand, November 8-13, 2011*, pages 1162–1170, 2011.

[3] Berend, Gábor. Sparse coding of neural word embeddings for multilingual sequence labeling. *Transactions of the Association for Computational Linguistics*, 5:247–261, 2017.

[4] Brooke, Julian, Snajder, Jan, and Baldwin, Timothy. Unsupervised acquisition of comprehensive multiword lexicons using competition in an n-gram lattice. *Transactions of the Association for Computational Linguistics*, 5:455–470, 2017.

[5] Brown, Peter F., deSouza, Peter V., Mercer, Robert L., Pietra, Vincent J. Della, and Lai, Jenifer C. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[6] Carpuat, Marine and Diab, Mona. Task-based evaluation of multiword expressions: A pilot study in statistical machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 242–245, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[7] Cholakov, Kostadin and Kordoni, Valia. Using word embeddings for improving statistical machine translation of phrasal verbs. In *Proceedings of the 12th Workshop on Multiword Expressions, MWE@ACL 2016, Berlin, Germany, August 11, 2016.*, 2016.

[8] Collobert, Ronan, Weston, Jason, Bottou, Léon, Karlen, Michael, Kavukcuoglu, Koray, and Kuksa, Pavel. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[9] Derczynski, Leon, Chester, Sean, and Bøgh, Kenneth. Tune your brown clustering, please. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 110–117. INCOMA Ltd. Shoumen, Bulgaria, 2015.

[10] Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[11] Kim, Su Nam. *Statistical Modeling of Multiword Expressions.* PhD thesis, University of Melbourne, Melbourne, 2008.

[12] Lafferty, John D., McCallum, Andrew, and Pereira, Fernando C. N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289. Morgan Kaufmann Publishers Inc., 2001.

[13] Liang, Percy. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology, 2005.

[14] Mairal, Julien, Bach, Francis, Ponce, Jean, and Sapiro, Guillermo. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 689–696. Association for Computing Machinery, 2009.

[15] Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[16] Okazaki, Naoaki. CRFsuite: a fast implementation of Conditional Random Fields (CRFs), 2007.

[17] Owoputi, Olutobi, Dyer, Chris, Gimpel, Kevin, Schneider, Nathan, and Smith, Noah A. Improved part-of-speech tagging for online conversational text with word clusters. In *In Proceedings of NAACL*, 2013.

[18] Pennington, Jeffrey, Socher, Richard, and Manning, Christopher. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics, 2014.

[19] Plank, Barbara, Søgaard, Anders, and Goldberg, Yoav. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418. Association for Computational Linguistics, 2016.

[20] Qu, Lizhen, Ferraro, Gabriela, Zhou, Liyuan, Hou, Weiwei, Schneider, Nathan, and Baldwin, Timothy. Big data small data, in domain out-of domain, known word unknown word: The impact of word representations on sequence labelling tasks. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 83–93. Association for Computational Linguistics, 2015.

[21] Ratinov, Lev and Roth, Dan. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 147–155. Association for Computational Linguistics, 2009.

[22] Sag, Ivan A., Baldwin, Timothy, Bond, Francis, Copestake, Ann, and Flickinger, Dan. Multiword Expressions: A Pain in the Neck for NLP. In *Proceedings of CICLing-2002*, pages 1–15, Mexico City, Mexico, 2002.

[23] Schneider, Nathan, Danchik, Emily, Dyer, Chris, and Smith, Noah A. Discriminative lexical semantic segmentation with gaps: Running the MWE gamut. *Transactions of the Association of Computational Linguistics*, 2:193–206, 2014.

[24] Stratos, Karl and Collins, Michael. Simple semi-supervised POS tagging. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 79–87. Association for Computational Linguistics, 2015.

[25] Tjong Kim Sang, Erik F. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158, 2002.

[26] Turian, Joseph, Ratinov, Lev, and Bengio, Yoshua. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394. Association for Computational Linguistics, 2010.

[27] Vincze, Veronika, Nagy, T. István, and Berend, Gábor. Detecting noun compounds and light verb constructions: A contrastive study. In *Proceedings of the Workshop on Multiword Expressions: From Parsing and Generation to the Real World*, MWE '11, pages 116–121, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[28] Vincze, Veronika, Nagy T., István, and Berend, Gábor. Multiword expressions and named entities in the Wiki50 corpus. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 289–295, Hissar, Bulgaria, September 2011. RANLP 2011 Organising Committee.

[29] Vincze, Veronika, T., István Nagy, and Zsibrita, János. Learning to detect English and Hungarian light verb constructions. *TSLP*, 10(2):6:1–6:25, 2013.

*Received 28th May 2018*

Acta Cybernetica **23** (2018) 815–828.

# Partition-Crossing Hypergraphs*

Csilla Bujtás[a] and Zsolt Tuza[ab]

**Abstract**

For a finite set $X$, we say that a set $H \subseteq X$ crosses a partition $\mathcal{P} = (X_1, \ldots, X_k)$ of $X$ if $H$ intersects $\min(|H|, k)$ partition classes. If $|H| \geq k$, this means that $H$ meets all classes $X_i$, whilst for $|H| \leq k$ the elements of the crossing set $H$ belong to mutually distinct classes. A set system $\mathcal{H}$ crosses $\mathcal{P}$, if so does some $H \in \mathcal{H}$. The minimum number of $r$-element subsets, such that every $k$-partition of an $n$-element set $X$ is crossed by at least one of them, is denoted by $f(n, k, r)$.

The problem of determining these minimum values for $k = r$ was raised and studied by several authors, first by Sterboul in 1973 [*Proc. Colloq. Math. Soc. J. Bolyai*, Vol. 10, Keszthely 1973, North-Holland/American Elsevier, 1975, pp. 1387–1404]. The present authors determined asymptotically tight estimates on $f(n, k, k)$ for every fixed $k$ as $n \to \infty$ [*Graphs Combin.*, 25 (2009), 807–816]. Here we consider the more general problem for two parameters $k$ and $r$, and establish lower and upper bounds for $f(n, k, r)$. For various combinations of the three values $n, k, r$ we obtain asymptotically tight estimates, and also point out close connections of the function $f(n, k, r)$ to Turán-type extremal problems on graphs and hypergraphs, or to balanced incomplete block designs.

**Keywords:** partition, set system, crossing set, Turán-type problem, hypergraph, upper chromatic number

## 1 Introduction

Let $X$ be a finite set. By a *$k$-partition* of $X$ we mean a partition $\mathcal{P} = (X_1, \ldots, X_k)$ into *precisely $k$ nonempty classes*. For a natural number $r \geq 2$, the family of all $r$-element subsets of $X$ — also termed *$r$-subsets*, for short (similarly, '$r$-set' may abbreviate '$r$-element set') — is denoted by $\binom{X}{r}$.A set system $\mathcal{H}$ over $X$ is *$r$-uniform* if $\mathcal{H} \subseteq \binom{X}{r}$.We shall use the term *hypergraph* for the pair $(X, \mathcal{H})$ — where $X$ is

*Research supported in part by the National Research, Development and Innovation Office – NKFIH under the grant SNN 116095.

[a]Faculty of Information Technology, University of Pannonia, H–8200 Veszprém, Egyetem u. 10, Hungary

[b]Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, H–1053 Budapest, Reáltanoda u. 13–15, Hungary

DOI: 10.14232/actacyb.23.3.2018.6

the set of *vertices* and $\mathcal{H}$ is the set of *edges* or *hyperedges* — and also for the set system $\mathcal{H}$ itself, when $X$ is understood. The number of vertices is called the *order* of $\mathcal{H}$, and will usually be denoted by $n$.

Given a $k$-partition $\mathcal{P} = (X_1, \ldots, X_k)$ of $X$, we say that an $r$-set $H \subseteq X$ *crosses* $\mathcal{P}$ if $H$ intersects $\min(r, k)$ partition classes. If $r \geq k$, this means that all classes $X_i$ are intersected, whilst for $r \leq k$ the elements of the crossing set $H$ belong to mutually distinct classes. A hypergraph $\mathcal{H}$ is said to cross $\mathcal{P}$ if so does at least one of its edges $H \in \mathcal{H}$.

It is a very natural problem to ask for the minimum number $f(n, k, r)$ of $r$-subsets (minimum number of edges in an $r$-uniform hypergraph), by which every $k$-partition of the $n$-element set $X$ is crossed. The importance of this question is demonstrated by the fact that its variants have been raised by several authors independently in different contexts under various names: Sterboul in 1973 [11] (*cochromatic number*, also discussed by Berge [4, pp. 151–152], Arocha et al. in 1992 [1] (*heterochromatic number*), and Voloshin in 1995 [14, p. 43, Open problem 11] (*upper chromatic number*, also recalled in the monograph [15, Chapter 2.6, p. 43, Problem 2]. What is more, the formula

$$f(n, 2, 2) = n - 1$$

is equivalent to the basic fact that every connected graph has at least $n - 1$ edges and that this bound is tight for all $n \geq 2$.

**Further terminology and notation.** For a family $\mathfrak{F}$ of $r$-uniform hypergraphs (or graphs if $r = 2$), and for any natural number $n$, we denote by $\mathrm{ex}(n, \mathfrak{F})$ the corresponding *Turán number*; that is, the maximum number of edges in an $r$-uniform hypergraph of order $n$ that does not contain any subhypergraph isomorphic to any $\mathcal{F} \in \mathfrak{F}$. If $\mathfrak{F}$ consists of just one hypergraph $\mathcal{F}$, we simply write $\mathrm{ex}(n, \mathcal{F})$ instead of $\mathrm{ex}(n, \{\mathcal{F}\})$.

An $r$-uniform hypergraph $(X, \mathcal{H})$ is *$r$-partite* if it admits a vertex partition $X_1 \cup \cdots \cup X_r = X$ such that $|H \cap X_i| = 1$ for all $H \in \mathcal{H}$ and all $1 \leq i \leq r$. If $\mathcal{H}$ consists of *all* $r$-sets meeting each $X_i$ in precisely one vertex, then we call it a *complete $r$-partite hypergraph*.

**Earlier results.** One can observe that a hypergraph crosses all 2-partitions of its vertex set if and only if it is connected. For this reason, beyond the equation $f(n, 2, 2) = n - 1$ mentioned above, we obtain that

$$f(n, 2, r) = \left\lceil \frac{n-1}{r-1} \right\rceil$$

because this is the minimum number of edges[1] in a connected $r$-uniform hypergraph of order $n$.

[1] It is well known that if $(X, \mathcal{H})$ is a *connected* hypergraph, then $\sum_{H \in \mathcal{H}} (|H| - 1) \geq |X| - 1$. The earliest source of this inequality that we have been able to find is Berge's classic book [3], where Proposition 4 on page 392 is stated more generally for a given number of connected components.

Let us observe further that the case of $r = 2$ simply means graphs with at most $k - 1$ connected components, therefore

$$f(n, k, 2) = n - k + 1.$$

This strong relationship with connected components, however, does not extend to $r > 2$.

As far as we know, for $k \geq 3$ and $r \geq 3$ only the 'diagonal case' $k = r$ of $f(n, k, r)$ has been studied up to now. Below we quote the known results, using the simplified notation $f(n, k)$ for $f(n, k, k)$.

- $f(n, k) \geq \frac{2}{n-k+2}\binom{n}{k}$, for every $n \geq k \geq 3$ ([12]; later proved independently in [1], and also rediscovered in [8]).
- $f(n, 3) = \lceil \frac{n(n-2)}{3} \rceil$, for every $n \geq 3$ ([7]; proved independently in a series of papers whose completing item is [2]; see also [13] for partial results).
- $f(n, n-2) = \binom{n}{2} - \mathrm{ex}(n, \{C_3, C_4\})$ holds[2] for every $n \geq 4$, where the last term is the Turán number for graphs of girth 5 ([12]).

Although the exact value of $f(n, k)$ is not known for any $k > 3$, its asymptotic behavior has been determined for quite a wide range of $k$.

**Theorem 1** ([5]). *Assume $n > k > 2$.*

*(i)* $f(n, k) \leq \frac{2}{n-1}\binom{n-1}{k} + \frac{n-1}{k-1}\left(\binom{n-2}{k-2} - \binom{n-k-1}{k-2}\right)$ *for all $n$ and $k$.*

*(ii)* $f(n, k) = (1 + o(1))\,\frac{2}{k}\binom{n-2}{k-1}$ *for all $k = o(n^{1/3})$ as $n \to \infty$.*

**Structure of the paper.** In Section 2, we first prove several preliminary results, also including an inequality for non-uniform partition-crossing hypergraphs in terms of the edge sizes. Then, we turn to uniform set systems and study the function $f(n, k, r)$ separately under the conditions $k \leq r$ and $r \leq k$. We prove general lower and upper bounds for $f(n, k, r)$ in both cases. In Section 3, we assume that $n - k$ and $n - r$ are fixed while $n \to \infty$, and give asymptotically tight estimates for $f(n, k, r)$. It is worth noting that the latter problem can be reduced to Turán-type problems if $k \leq r$, while the same question leads us to the theory of balanced incomplete block designs if $r \leq k$ is assumed.

## 2 General estimates

Most of this section deals with uniform hypergraphs; but we shall also put comments on non-uniform ones which cross either all partitions or at least some large families of partitions. Nevertheless the uniform systems play a central role in partition crossing, what will turn out already in the next subsection.

[2] It was quoted with a misprint in the paper [5].

## 2.1 Monotonicity

**Proposition 2.** *For every three integers $r, k, k'$, if either*

(*i*) $2 \le r \le k \le k' \le n$, *or*

(*ii*) $2 \le k' \le k \le r \le n$

*holds, and an $r$-uniform hypergraph $\mathcal{H}$ crosses all $k$-partitions of the vertex set, then $\mathcal{H}$ crosses all $k'$-partitions, as well. As a consequence, for every four integers $n, k, k', r$ satisfying (i) or (ii) we have*

$$f(n,k,r) \ge f(n,k',r).$$

**Proof** Assume that an $r$-uniform hypergraph $\mathcal{H}$ crosses all $k$-partitions of the vertex set $X$. Consider a $k'$-partition $\mathcal{P}' = (X_1, \ldots, X_{k'})$ of $X$.

(*i*) If $r \le k \le k'$, take the union of the last $k' - k + 1$ partition classes of $\mathcal{P}'$. Due to our assumption, $\mathcal{H}$ crosses the $k$-partition $\mathcal{P} = (X_1, \ldots, X_{k-1}, \bigcup_{i=k}^{k'} X_i)$ obtained. Since $r \le k$, this means that there exists an $H \in \mathcal{H}$ which contains at most one element from each partition class of $\mathcal{P}$. Hence, the same $H$ and consequently, $\mathcal{H}$ as well, crosses the $k'$-partition $\mathcal{P}'$.

(*ii*) Next, assume that $k' \le k \le r$ holds. Since the statement clearly holds for $k' = k$, we may suppose $k' < k \le n$. Then, some of the $k'$ partition classes of $\mathcal{P}'$ can be split into nonempty parts such that a $k$-partition $\mathcal{P}$ is obtained. By assumption, some $H \in \mathcal{H}$ crosses $\mathcal{P}$. This means that the $r$-element $H$ contains at least one element from each partition class. By the construction of $\mathcal{P}$, $H$ contains at least one element from every partition class of $\mathcal{P}'$; that is, $\mathcal{H}$ crosses $\mathcal{P}'$.

Since the above arguments are valid for any $k'$-partition $\mathcal{P}'$, the statements follow. □

The analogous property is valid for the other parameter of $f(n,k,r)$ as well.

**Proposition 3.** *For every four integers $n, k, r, r'$, if*

(*i*) $2 \le r' \le r \le k \le n$, *or*

(*ii*) $2 \le k \le r \le r' \le n$ *holds, then*

$$f(n,k,r) \ge f(n,k,r').$$

**Proof** Consider an $r$-uniform hypegraph $(X, \mathcal{H})$ of size $f(n,k,r)$ which crosses all $k$-partitions of the $n$-element vertex set $X$.

(*i*) If $r' \le r \le k$, then for each $H \in \mathcal{H}$ choose an $r'$-element subset $H'$ and define the $r'$-uniform set system $\mathcal{H}' = \{H' \mid H \in \mathcal{H}\}$. Since for every $k$-partition $\mathcal{P}$ there exists an $H \in \mathcal{H}$ which contains at most one element from each partition class, the same is true for the corresponding $H' \in \mathcal{H}'$. Hence, $\mathcal{H}'$ crosses all $k$-partitions and has at most $f(n,k,r)$ elements. This proves that $f(n,k,r) \ge f(n,k,r')$.

(*ii*) In the other case we have $k \le r \le r'$. Let each $H \in \mathcal{H}$ be extended to an arbitrary $r'$-element $H'$. We observe that the $r'$-uniform set system $\mathcal{H}' = \{H' \mid$

$H \in \mathcal{H}\}$ has at most $f(n,k,r)$ elements and crosses all $k$-partitions. Indeed, for every $k$-partition $\mathcal{P}$, there exists some $H \in \mathcal{H}$ intersecting each partition class of $\mathcal{P}$, and hence the same is true for the corresponding $H' \in \mathcal{H}'$. This yields again that $f(n,k,r) \geq f(n,k,r')$ is valid. □

The following corollaries show the central role of the 'symmetric' case $k = r$:

**Corollary 4.** *If an $r$-uniform hypergraph $\mathcal{H}$ crosses all $r$-partitions of the vertex set $X$, then $\mathcal{H}$ crosses all partitions of $X$.*

Numerically, we have obtained that the function $f_{n,r}(x) = f(n,x,r)$ (where $x$ is an integer in the range $2 \leq x \leq n$) has its maximum value when $x = r$; and the situation is similar if $n$ and $k$ are fixed and $r$ is variable; that is, the function $f_{n,k}(x) = f(n,k,x)$ attains its maximum at $x = k$.

**Corollary 5.** *For every three integers $n \geq k, r \geq 2$,*

$$f(n,k,r) \leq f(n,k,k).$$

**Corollary 6.** *For every three integers $n \geq k, r \geq 2$,*

$$f(n,k,r) \leq f(n,r,r).$$

## 2.2 Lower bound for non-uniform systems

For hypergraphs without very small edges, we prove the following general inequality.

**Theorem 7.** *Let $k \geq 2$ be an integer, and let $(X,\mathcal{H})$ be a hypergraph of order $n$, which contains no edge $H \in \mathcal{H}$ of cardinality smaller than $k$. If $\mathcal{H}$ crosses all $k$-partitions of $X$, then*

$$\sum_{H \in \mathcal{H}} \binom{|H|}{k} \frac{1}{|H|-k+2} \geq \binom{n}{k} \frac{1}{n-k+2}.$$

**Proof** Since $|H| \geq k$ holds for every $H \in \mathcal{H}$, a $k$-partition $\mathcal{P}$ of $X$ is crossed by $\mathcal{H}$ if, and only if, there exists an edge in $\mathcal{H}$ which intersects all the $k$ partition classes of $\mathcal{P}$. For every $(k-2)$-element subset $Y = \{x_1, \ldots, x_{k-2}\}$ of $X$, define

$$\mathcal{H}_Y^- = \{A \mid A \subseteq (X \setminus Y) \ \wedge \ (A \cup Y) \in \mathcal{H}\}.$$

We claim that $\mathcal{H}_Y^-$ is connected on $X \setminus Y$. Assume for a contradiction that it is not, and denote one of its components by $Z$. Consider the $k$-partition

$$\{x_1\}, \ \ldots, \ \{x_{k-2}\}, \ Z, \ X \setminus (Y \cup Z)$$

This is not crossed by $\mathcal{H}$ since a crossing set $H$ would contain all of $x_1, \ldots, x_{k-2}$, moreover at least one element from each of the last two partition classes, what contradicts to our assumption on disconnectivity.

Therefore, $\mathcal{H}_Y^-$ must be connected on the $(n-k+2)$-element $X \setminus Y$, and hence

$$\sum_{A \in \mathcal{H}_Y^-} (|A|-1) \geq (n-k+2)-1.$$

The corresponding inequality holds for every $Y \in \binom{X}{k-2}$. Moreover, for each edge $H \in \mathcal{H}$, every $(|H|-k+2)$-element subset of $H$ is counted in exactly one of these $\binom{n}{k-2}$ inequalities. Hence, we have

$$\sum_{H \in \mathcal{H}} \binom{|H|}{k-2}(|H|-k+1) \geq \binom{n}{k-2}(n-k+1),$$

which is equivalent to the assertion. □

Beside the rather trivial hypergraph with vertex set $X$ and edge set $\mathcal{H} = \{X\}$, which crosses every partition of $X$, the following construction also shows that Theorem 7 is tight.

**Example 8.** Let $n = |X| = 2m$ be even. Let the edge set of $\mathcal{H}$ consist of one $m$-subset $H$ of $X$ together with $m$ mutually disjoint 2-element sets, each of which has precisely one vertex in $H$ and one in $X \setminus H$. This hypergraph crosses all partitions of $X$. Indeed, if none of the $m$ selected 2-sets crosses a partition $\mathcal{P}$, then each class of $\mathcal{P}$ meets $H$. For this $\mathcal{H}$, both sides of the inequality in Theorem 7 equal $\frac{n-1}{2}$ for $k = 2$. (We necessarily have $k = 2$, due to the conditions in the theorem.)

## 2.3 Estimates for $k \leq r$

The following lower bound follows immediately from Theorem 7.

**Corollary 9.** *For every three integers $n \geq r \geq k \geq 2$ the inequality*

$$f(n,k,r) \geq \frac{\binom{n}{k}}{\binom{r}{k}} \cdot \frac{r-k+2}{n-k+2}$$

*holds.*

Next, we prove a general asymptotic upper bound.

**Proposition 10.** *For every two fixed integers $r \geq k \geq 2$ the inequality*

$$f(n,k,r) \leq \frac{\binom{n}{k}}{\binom{r}{k}} \cdot \frac{r}{n} + o(n^{k-1})$$

*holds as $n \to \infty$.*

**Proof** If $k = r$, then the inequality holds also without the error term, and as a matter of fact, an even better upper bound on $f(n,r,r)$ is guaranteed by Theorem 1($i$). Hence, we may suppose $r > k$.

Consider an $n$-element vertex set $X = X' \cup \{z\}$ and an $(r-1)$-uniform hypergraph $\mathcal{H}'$ over $X'$ such that every $(k-1)$-subset of $X'$ is covered by at least one $H' \in \mathcal{H}'$. By Rödl's theorem [10], such hypergraphs $\mathcal{H}'$ of size

$$|\mathcal{H}'| = \frac{\binom{n-1}{k-1}}{\binom{r-1}{k-1}} + o(n^{k-1})$$

exist as $n \to \infty$.

Consider now the $r$-uniform hypergraph

$$\mathcal{H} = \{H' \cup \{z\} \mid H' \in \mathcal{H}'\}.$$

For every $k$-partition $\mathcal{P}$ we can choose a $k$-element crossing set $A$ with $z \in A$, by picking any vertex from each of those classes of $\mathcal{P}$ which do not contain $z$. Since $A \setminus \{z\} \subset H'$ for some $H' \in \mathcal{H}'$, it follows that $\mathcal{H}$ crosses $\mathcal{P}$. □

We note that, beyond tight asymptotics, the above construction can be applied also to derive exact results for some restricted combinations of the parameters.

Next, we establish recursive relations to get lower bounds on $f(n,k,r)$. Although they do not improve earlier bounds automatically, such inequalities may raise the possibility to propagate better estimates for larger values of the parameters when they are available for smaller ones.

**Proposition 11.** *If $n \geq r \geq r' \geq k \geq 2$, then*

$$f(n,k,r) \geq \frac{f(n,k,r')}{f(r,k,r')}.$$

**Proof** Given an $n$-element vertex set $X$, consider an $r$-uniform hypergraph $\mathcal{H}$ of size $f(n,k,r)$ which crosses all $k$-partitions. Then, for each $H_j \in \mathcal{H}$ construct an $r'$-uniform hypergraph $\mathcal{H}'_j$ crossing all $k$-partitions of the set $H_j$. This can be done such that $|\mathcal{H}'_j| = f(r,k,r')$, hence the $r'$-uniform $\mathcal{R} = \bigcup_{j=1}^{f(n,k,r)} \mathcal{H}'_j$ contains at most $f(n,k,r) \cdot f(r,k,r')$ sets.

For every partition $\mathcal{P} = (X_1, \ldots, X_k)$, there exists some $H_j \in \mathcal{H}$ with $|X_i \cap H_j| \geq 1$ for every $1 \leq i \leq k$. Moreover, for this $j$, the system $\mathcal{H}'_j$ crosses also the $k$-partition $X_1 \cap H_j, \ldots, X_k \cap H_j$. Consequently, there exists an $R \in \mathcal{H}'_j \subseteq \mathcal{R}$ which intersects every class of $\mathcal{P}$. Thus, $\mathcal{R}$ crosses all $k$-partitions of $X$, therefore

$$f(n,k,r) \cdot f(r,k,r') \geq f(n,k,r')$$

holds and the theorem follows. □

Particularly, if $r'$ is chosen to be equal to $k$, we obtain that

$$f(n,k,r) \geq \frac{f(n,k)}{f(r,k)}.$$

Since $f(k+1,k)=k$, then

$$f(n,k,k+1) \geq \frac{f(n,k)}{k}.$$

More generally, applying Proposition 11 repeatedly, and using the fact $f(i,k,i-1)=k$ that is valid for all $i>k$ (cf. Proposition 19 below), we obtain the following lower bound.

**Corollary 12.** *If $n \geq r \geq k \geq 2$, then*

$$f(n,k,r) \geq \frac{f(n,k)}{\prod_{i=k+1}^{r} f(i,k,i-1)} = \frac{f(n,k)}{k^{r-k}}.$$

## 2.4 Estimates for $k \geq r$

**Proposition 13.** *For every three integers $n \geq k \geq r \geq 2$ the inequality*

$$f(n,k,r) \geq \frac{\binom{n}{r-1}}{\binom{k-2}{r-2}} \cdot \frac{n-k+2}{r(n-r+2)}$$

*holds.*

**Proof** Consider an $r$-uniform hypergraph $\mathcal{H}$ on the $n$-element vetex set $X$, such that $\mathcal{H}$ crosses all $k$-partitions. We claim that every $(k-1)$-subset of $X$ shares at least $r-1$ vertices with some $H \in \mathcal{H}$. Suppose for a contradiction that a set $A \in \binom{X}{k-1}$ intersects no $H \in \mathcal{H}$ in more than $r-2$ elements. Then every $H \in \mathcal{H}$ has at least two vertices in $X \setminus A$. Now, consider the $k$-partition whose first partition class is $X \setminus A$ and the others are singletons. This partition is not crossed by $\mathcal{H}$, which is a contradiction.

Consequently, every $(k-1)$-element subset of $X$ must contain an $(r-1)$-element subset of some $H \in \mathcal{H}$. Hence, for the 'shadow' system

$$\partial_{r-1} = \left\{ B \mid \exists H \in \mathcal{H} \quad \text{s.t. } B \in \binom{H}{r-1} \right\},$$

the independence number must be smaller than $k-1$. Taking into consideration the lower bound on the complementary Turán number $T(n,k-1,r-1) = \binom{n}{r-1} - \mathrm{ex}\,(n, \mathcal{K}_{k-1}^{(r-1)})$ of complete uniform hypergraphs, as proved in [6],

$$r \cdot |\mathcal{H}| \geq |\partial_{r-1}| \geq T(n,k-1,r-1) \geq \frac{\binom{n}{r-1}}{\binom{k-2}{r-2}} \cdot \frac{n-k+2}{n-r+2}$$

is obtained, from which the statement follows. □

For $k$ and $r$ fixed, the lower bound gives the right order $O(n^{r-1})$, as shown by the following construction.

**Theorem 14.** *Let $k \geq 3$, and assume that $k-2$ is divisible by $r-2$. If $n \to \infty$, then*

$$f(n,k,r) \leq \frac{2(r-2)^{r-2}}{r(k-2)^{r-2}} \binom{n}{r-1} + o(n^{r-1}).$$

**Proof** Let $|X| = n$, denote $q = (k-2)/(r-2)$, and write $n' = \lceil (n-1)/q \rceil + 1$. We fix a special element $z \in X$, and partition the remaining $(n-1)$-element set $X \setminus \{z\}$ into $q$ nearly equal parts, the largest one having $n'-1$ vertices:

$$X = Y_1 \cup \cdots \cup Y_q \cup \{z\}, \qquad |Y_i| = \left\lfloor \frac{n+i-2}{q} \right\rfloor \qquad \text{for all } \ 1 \leq i \leq q.$$

For every set $Y_i \cup \{z\}$ we take an optimal $r$-uniform hypergraph $\mathcal{H}_i$ crossing all $r$-partitions. By Theorem 1, we have

$$|\mathcal{H}_i| \leq f(n',r) \leq (1+o(1))\,\frac{2}{r}\binom{n'-2}{r-1}.$$

Here $n'-2 < n/q = \frac{r-2}{k-2}\,n$, hence the binomial coefficient on the right-hand side is smaller than $\left(\frac{r-2}{k-2}\right)^{r-1}\binom{n}{r-1}$ Let $\mathcal{H} = \mathcal{H}_1 \cup \cdots \cup \mathcal{H}_q$. By the estimates above, we have

$$|\mathcal{H}| \leq \frac{2(r-2)^{r-2}}{r(k-2)^{r-2}} \binom{n}{r-1} + o(n^{r-1})$$

as $n \to \infty$. To complete the proof, it suffices to show that $\mathcal{H}$ crosses all $k$-partitions of $X$.

Let $\mathcal{P}$ be any partition into $k = 1 + q(r-2) + 1$ classes. One of the classes contains $z$. By the pigeonhole principle, there is an index $i$ $(1 \leq i \leq q)$ such that, among the other $k-1$ classes of $\mathcal{P}$ there exist at least $r-1$ which have at least one vertex in $Y_i$. Hence we have a partition $\mathcal{P}_i$ induced on $Y_i \cup \{z\}$, with some number $r'$ of classes, where $r' \geq r$. Since the $r$-uniform $\mathcal{H}_i$ crosses all $r$-partitions of $Y_i \cup \{z\}$, Corollary 4 implies that $\mathcal{H}_i$ crosses $\mathcal{P}_i$, too. That is, an $r$-set $H_i \in \mathcal{H}_i$ has all its vertices in mutually distinct classes of $\mathcal{P}_i$, which are then in distinct classes of $\mathcal{P}$ as well. Thus, $\mathcal{H}$ crosses $\mathcal{P}$. □

The idea behind the construction of the above proof also yields the following additive upper bound.

**Proposition 15.** *Suppose that all the following conditions hold:*

- $n \geq k \geq r$,
- $n \leq 1 - p + \sum_{i=1}^{p} n_i$,
- $k \leq 2 - 2p + \sum_{i=1}^{p} k_i$,
- $n_i \geq k_i \geq r$ *for every* $1 \leq i \leq p$.

*Then*

$$f(n,k,r) \leq \sum_{i=1}^{p} f(n_i,k_i,r).$$

## 3 Asymptotics for large $k$ and $r$

In this section we prove asymptotically tight estimates for $f(n,k,r)$, under the assumptions that the differences $s=n-k$ and $t=n-r$ are fixed and $n\to\infty$. For this purpose, we need to consider two types of complementation — one from the viewpoint of set theory, the other one analogously to graph theory.

- Given a hypergraph $(X,\mathcal{H})$, let $(X,\mathcal{H}^c)$ denote the hypergraph of the complements of the edges. That is, $\mathcal{H}^c=\{X\setminus H \mid H\in\mathcal{H}\}$.
- Given an *$r$-uniform* hypergraph $(X,\mathcal{H})$, its complement $\overline{\mathcal{H}}$ contains all the $r$-element subsets of $X$ which are missing from $\mathcal{H}$. Formally, $\overline{\mathcal{H}}=\binom{X}{r}\setminus\mathcal{H}$.

**Theorem 16.** *Let $s$ and $t$ be fixed, with $s\le t$, and $n\to\infty$. Then*

$$f(n,n-s,n-t)=(1+o(1))\frac{\binom{n}{s}}{\binom{t}{s}}.$$

**Proof** First we prove the lower bound $f(n,n-s,n-t)\ge(1-o(1))\binom{n}{s}/\binom{t}{s}$. Suppose for a contradiction that there exists a constant $\epsilon>0$ and an infinite sequence of $r$-uniform hypergraphs $(X,\mathcal{H})$ with $n$ vertices and $m$ edges, edge size $r=n-t$, such that $\mathcal{H}$ crosses all $(n-s)$-partitions of its $n$-element vertex set $X$, but $m\le\frac{\binom{n}{s}}{\binom{t}{s}}-\epsilon n^s$. We consider the $t$-uniform hypergraph $\mathcal{H}^c$ whose edges are the complements of the edges of $\mathcal{H}$. Since it has $m$ edges, there are at least $\epsilon\binom{t}{s}n^s\ge Cn^s$ distinct $s$-tuples of $X$ not covered by the edges of $\mathcal{H}^c$. Note that $C$ can be chosen as a positive absolute constant, valid for all possible values of $n$, once we fix the triplet $s,t,\epsilon$. We let $\mathcal{F}$ to be the collection of $s$-tuples not contained in any of the edges of $\mathcal{H}^c$. Hence $|\mathcal{F}|\ge Cn^s$.

Consider now the complete $s$-partite hypergraph $\mathcal{F}_s$ on $2s$ vertices, each partite set having just 2 vertices. That is, the vertex set of $\mathcal{F}_s$ is $V_1\cup\cdots\cup V_s$, with $|V_i|=2$ for all $1\le i\le s$, and an $s$-element set $F$ is an edge in $\mathcal{F}_s$ if and only $|F\cap V_i|=1$ for every $i$. It is well known that the Turán number of $\mathcal{F}_s$ satisfies

$$\mathrm{ex}\,(n,\mathcal{F}_s)=o(n^s)$$

for any fixed $s$, as $n\to\infty$. Thus, if $n$ is chosen to be sufficiently large, $\mathcal{F}$ contains a subhypergraph $\mathcal{F}'$ isomorphic to $\mathcal{F}_s$.

We now consider the partition $\mathcal{P}$ of $X$ into $k=n-s$ classes in which the $s$ partite sets of $\mathcal{F}'$ are 2-element classes, and the other $n-2s$ classes are singletons. By assumption, $\mathcal{H}$ crosses $\mathcal{P}$. It means that there exists an edge $H\in\mathcal{H}$ that meets each of the 2-element classes in at most one vertex. Let $x_i$ be a vertex in $V_i\setminus H$ for $i=1,\dots,s$. Then $\{x_1,\dots,x_s\}\notin\mathcal{F}$, which is a contradiction to $\{x_1,\dots,x_s\}\in\mathcal{F}_s\subset\mathcal{F}$, hence completing the proof of the lower bound.

Next, we prove the upper bound $f(n,n-s,n-t)\le(1+o(1))\binom{n}{s}/\binom{t}{s}$. For every $n$, consider a $t$-uniform hypergraph $\mathcal{H}_n^0$ on the $n$-element vertex set $X$, such that

each $s$-subset of $X$ is contained in a $t$-set $H \in \mathcal{H}_n^0$. By Rödl's theorem [10], if $s$ and $t$ are fixed and $n \to \infty$, then $\mathcal{H}_n^0$ can be chosen such that $|\mathcal{H}_n^0| = \binom{n}{s}/\binom{t}{s} + o(n^s)$.

Starting with such a system $\mathcal{H}_n^0$, we consider the hypergraph $\mathcal{H}_n = (\mathcal{H}_n^0)^c$ whose edge set is $\{X \setminus H \mid H \in \mathcal{H}_n^0\}$. By the complementation, for $k = n-s$ and $r = n-t$, each $k$-element subset of $X$ contains some $r$-element set $H \in \mathcal{H}_n$. Then, for any $k$-partition $\mathcal{P}$ of $X$, we can pick one vertex from each partition class, and this $k$-element set has to contain an edge $H \in \mathcal{H}_n$. Hence, $\mathcal{H}_n$ crosses all $k$-partitions of the vertex set, moreover we have $|\mathcal{H}_n| = |\mathcal{H}_n^0|$. This yields the claimed upper bound on $f(n, n-s, n-t)$. □

In particular, for $s = t$ we have the following consequence. We formulate it for $s \geq 2$, because the case of $f(n,n,n) = 1$ is trivial and the exact formula of $f(n, n-1, n-1) = n-1$ is a particular case of Proposition 19 below.

**Corollary 17.** *For every $s \geq 2$, as $n \to \infty$*

$$f(n, n-s, n-s) = \binom{n}{s} + o(n^s).$$

To study the other range for $f(n, n-s, n-t)$, namely $s > t$, first we will make a simple but useful observation. We say that a set $T$ is a *transversal* of a partition[3] $\mathcal{P} = (X_1, \ldots, X_k)$ if $|T \cap X_i| \geq 1$ holds for every $i$. The complement $S = X \setminus T$ of a transversal $T$ is called an *independent set* for $\mathcal{P}$. This means that $|S \cap X_i| < |X_i|$ holds for every partition class. Let $\mathcal{I}_t(\mathcal{P})$ denote the set system containing all $t$-element independent sets for the partition $\mathcal{P}$.

**Proposition 18.** *Let $(X, \mathcal{H})$ be an $r$-uniform hypergraph with $|X| = n$, and assume that $k \leq r$. Then, $\mathcal{H}$ crosses all $k$-partitions of the vertex set $X$ if and only if for every $k$-partition $\mathcal{P}$ of $X$ we have $\mathcal{I}_{n-r}(\mathcal{P}) \not\subseteq \overline{\mathcal{H}^c}$.*

**Proof** For a given $k$-partition $\mathcal{P}$, $\mathcal{H}$ is crossing if and only if it contains a transversal $T$ for $\mathcal{P}$; that is, if $\mathcal{H}^c$ contains an $(n-r)$-element independent set for $\mathcal{P}$. This equivalently means that $\overline{\mathcal{H}^c}$ does not contain all elements of $\mathcal{I}_{n-r}(\mathcal{P})$. Consequently, $\mathcal{H}$ crosses all $k$-partitions if and only if for every $k$-partition $\mathcal{P}$, $\overline{\mathcal{H}^c}$ does not contain $\mathcal{I}_{n-r}(\mathcal{P})$ as a subsystem. □

Concerning $f(n, n-s, n-t)$ the case of $t = 1$ is very simple. Certainly $s = 0$ means that all partition classes are singletons, hence $f(n,n,r) = 1$ for all values of $r \leq n$, also including $r = n-1$. The situation for smaller $k$ is different.

**Proposition 19.** *For every $n > k \geq 1$, we have $f(n, k, n-1) = k$.*

**Proof** For $X = \{x_1, \ldots, x_n\}$ define $\mathcal{H} = \{X \setminus \{x_i\} \mid 1 \leq i \leq k\}$. Consider any $k$-partition $\mathcal{P}$. It either has a class with at least two vertices $x_i, x_j$ in the range

[3] In fact this is the same as a transversal (also called vertex cover or hitting set) of the hypergraph $(X, \{X_1, \ldots, X_k\})$ in which the classes $X_i$ of the partition are viewed as edges. This also justifies the term 'independent set' for the complementary notion.

$1 \le i < j \le k$, or a class containing both $x_n$ and some $x_i$ with $1 \le i \le k$. Then we can choose $X \setminus \{x_i\} \in \mathcal{H}$, which crosses $\mathcal{P}$. Consequently, $f(n,k,n-1) \le k$.

To see the reverse inequality $f(n,k,n-1) \ge k$, without loss of generality we may restrict our attention to the $(n-1)$-uniform hypergraph $\mathcal{H}^- = \{X \setminus \{x_i\} \mid 1 \le i \le k-1\}$ which represents all $(n-1)$-uniform ones with $k-1$ edges up to isomorphism. Then the partition

$$\{x_1\},\ \ldots,\ \{x_{k-1}\},\ \{x_k, x_{k+1}, \ldots, x_n\}$$

is not crossed by any $H \in \mathcal{H}^-$, thus $k-1$ edges are not enough. □

The problem becomes more complicated for $t > 1$. First we consider the case of $r = n-2$, and then a general estimate for $k = n-s \le n-t = r$ will be given under the assumption that $s$ and $t$ are fixed.

**Proposition 20.** *For every fixed $s \ge 2$,*

*(i)* $f(n, n-s, n-2) = \binom{n}{2} - \mathrm{ex}(n, \{K_{s+1}, K_{2s} - sK_2\})$;

*(ii)* $f(n, n-s, n-2) = \frac{1}{2s-2}\, n^2 + o(n^2)$, *if* $n \to \infty$.

**Proof** Consider a graph $G = (V, F)$ of order $n$, which contains neither a complete graph $K_{s+1}$ of order $s+1$, nor a complete graph minus a perfect matching $K_{2s} - sK_2$ on $2s$ vertices. By the double complementation we obtain the $(n-2)$-uniform hypergraph $(V, \mathcal{H}) = (\overline{G})^c$ with vertex set $V$ and edge set

$$\mathcal{H} = \{V \setminus e \mid e \in \binom{V}{2} \ \wedge\ e \notin F\}.$$

We claim that $\mathcal{H}$ crosses all $(n-s)$-partitions of $V$.

First, consider a partition $\mathcal{P} = (X_1, X_2, \ldots, X_{n-s})$ with at least one partition class $|X_i| \ge 3$. We can assume without loss of generality that $|X_1| \ge 3$. We also consider the partition $\mathcal{P}'$, obtained by removing all but one vertex from each of $X_2, \ldots, X_{n-s}$ and putting these vertices into $X_1$. This $\mathcal{P}'$ has an $(s+1)$-element class $X_1'$ and further $n-s-1$ singleton classes. Since the class $X_1$ in $\mathcal{P}$ has more than two vertices, every edge of $\mathcal{H}$ meets $X_1$. Hence, the hypergraph $\mathcal{H}$ does not cross $\mathcal{P}$ if and only if each of its edges is disjoint from at least one of the classes $X_2, \ldots, X_{n-s}$. But then every edge is also disjoint from at least one of the singleton classes of $\mathcal{P}'$, and so $\mathcal{H}$ does not cross $\mathcal{P}'$ either.

Consequently, it is sufficient to ensure that $\mathcal{H}$ crosses all $(n-s)$-partitions with classes of cardinalities $(s+1, 1, \ldots, 1)$ and $(2, \ldots, 2, 1, \ldots 1)$, and this will imply that $\mathcal{P}$ crosses all $(n-s)$-partitions.

An $(n-2)$-uniform hypergraph $\mathcal{H}$ crosses every partition of type $(s+1, 1, \ldots, 1)$ if, and only if, for every $(s+1)$-element subset $S$ of $V$, there exits an edge $H \in \mathcal{H}$ with $|H \cap S| = s-1$; that is, $\mathcal{H}^c$ has an edge inside $S$, and equivalently, $G = \overline{\mathcal{H}^c}$ contains no complete subgraph $K_{s+1}$. For the other case, $\mathcal{H}$ crosses every partition of type $(2, \ldots, 2, 1, \ldots 1)$, if and only if for every $s$ disjoint pairs of vertices there

exists an edge $H$ whose complement $\overline{H}$ contains two vertices from different pairs. This exactly means that $G = \overline{\mathcal{H}^c}$ does not contain a subgraph $K_{2s} - sK_2$.

Consequently, an $(n-2)$-uniform $\mathcal{H}$ crosses all $(n-s)$-partitions if and only if $G$ is $(K_{s+1}, K_{2s} - sK_2)$-free. Applying the Erdős–Stone Theorem [9], for $s \geq 3$ this yields

$$f(n, n-s, n-2) = \binom{n}{2} - \mathrm{ex}(n, \{K_{s+1}, K_{2s} - sK_2\})$$

$$= \binom{n}{2} - (1+o(1)) \cdot \mathrm{ex}(n, K_s) = \frac{1}{2s-2}\, n^2 + o(n^2).$$

In fact the asymptotic formula is valid also for $s = 2$ because then the exclusion of $K_{2,2} \cong C_4$ implies that $\mathrm{ex}(n, \{K_{s+1}, K_{2s} - sK_2\}) = o(n^2)$. □

**Theorem 21.** *Let $s$ and $t$ be fixed, with $s > t \geq 2$, and $n \to \infty$. Then,*

$$f(n, n-s, n-t) \leq (1-c)\binom{n}{t}$$

*for some constant $c = c(s,t) > 0$.*

**Proof** Let $\mathcal{H}_t$ be the complete $t$-partite hypergraph with vertex set $X_1 \cup \cdots \cup X_t$ such that each partite class has cardinality $|X_i| = \lfloor n/t \rfloor$ or $|X_i| = \lceil n/t \rceil$. We have $|\mathcal{H}_t| = (1-o(1))\,(n/t)^t$ as $n \to \infty$, hence there exists a universal constant $c = c(t) > 0$ such that $|\mathcal{H}_t| \geq c\binom{n}{t}$ for all $n > t$. Let $\mathcal{H} = \big(\overline{\mathcal{H}_t}\big)^c$. Then $|\mathcal{H}| \leq (1-c)\binom{n}{t}$.

We claim that $\mathcal{H}$ crosses all $(n-s)$-partitions whenever $s > t$. Indeed, let $\mathcal{P}$ be any $(n-s)$-partition of $X$. Consider an $s$-set $S$ obtained by deleting precisely one vertex from each class of $\mathcal{P}$. Since $s > t$, this $S$ contains two vertices from the same class of $\mathcal{H}_t$, say $x', x'' \in X_i$. Therefore we can take a $t$-subset $T \subset S$ containing both $x'$ and $x''$, consequently $T \notin \mathcal{H}_t$. Thus, $X \setminus T \in \mathcal{H}$ holds, and this $X \setminus T$ meets all classes of $\mathcal{P}$ because it contains all elements of $X \setminus S$. It follows that $\mathcal{H}$ crosses every $\mathcal{P}$, hence

$$f(n, n-s, n-t) \leq |\mathcal{H}| \leq (1-c)\binom{n}{t}.$$

□

# References

[1] J. L. Arocha, J. Bracho, and V. Neumann-Lara, On the minimum size of tight hypergraphs. *J. Graph Theory* **16** (1992) 319–326.

[2] J. L. Arocha and J. Tey, The size of minimum 3-trees. *J. Graph Theory* **54** (2007) 103–114.

[3] C. Berge, *Graphs and Hypergraphs.* (North-Holland, 1973)

[4] C. Berge, *Hypergraphs.* (North-Holland, 1989)

[5] Cs. Bujtás and Zs. Tuza, Smallest set-transversals of $k$-partitions. *Graphs Combin.* **25** (2009) 807–816.

[6] D. de Caen, Extension of a theorem of Moon and Moser on complete subgraphs, *Ars Combinatoria* **16** (1983) 5–10.

[7] K. Diao, G. Liu, D. Rautenbach, and P. Zhao, A note on the least number of edges of 3-uniform hypergraphs with upper chromatic number 2. *Discrete Math.* **306** (2006) 670–672.

[8] K. Diao, P. Zhao, and H. Zhou, About the upper chromatic number of a $C$-hypergraph. *Discrete Math.* **220** (2000) 67–73.

[9] P. Erdős and A. H. Stone, On the structure of linear graphs. *Bull. Amer. Math. Soc.* **52** (1946) 1087–1091.

[10] V. Rödl, On a packing and covering problem. *Europ. J. Combin.* **5** (1985) 69–78.

[11] F. Sterboul, A new combinatorial parameter. In: *Infinite and Finite Sets* (A. Hajnal et al., eds.), Colloq. Math. Soc. J. Bolyai, **10**, Vol. III, Keszthely 1973 (North-Holland/American Elsevier, 1975) 1387–1404.

[12] F. Sterboul, Un problème extrémal pour les graphes et les hypergraphes. *Discrete Math.* **11** (1975) 71–78.

[13] F. Sterboul, A problem in constructive combinatorics and related questions. In: *Combinatorics* (A. Hajnal and V. T. Sós, eds.), Colloq. Math. Soc. J. Bolyai, **18**, Vol. II, Keszthely 1976 (North-Holland, 1978) 1049–1064.

[14] V. Voloshin, On the upper chromatic number of a hypergraph. *Australasian J. Combin.* **11** (1995) 25–45.

[15] V. I. Voloshin, *Coloring Mixed Hypergraphs: Theory, Algorithms and Applications.* Fields Institute Monographs **17** (AMS, 2002)

*Received 27th February 2018*

Acta Cybernetica **23** (2018) 829–852.

# Approximations to the Normal Probability Distribution Function using Operators of Continuous-valued Logic

József Dombi[a] and Tamás Jónás[b]

**Abstract**

In this study, novel approximation methods to the standard normal probability distribution function are introduced. The techniques presented are founded on applications of certain operators of continuous-valued logic. It is demonstrated here that application of the averaging Dombi conjunction operator to two symmetric Sigmoid fuzzy membership functions results in a function that is identical with Tocher's approximation to the standard normal probability distribution function. Next, an approximation connected with a unary fuzzy modifier operator is discussed. Namely, the so-called Kappa function is applied for constructing a novel probability distribution function. It is shown here that the asymptotic Kappa function is just the Sigmoid function and the proposed Quasi Logistic probability distribution function can be utilized to approximate the standard normal probability distribution function. It is also explained how the new probability distribution function is connected with the generator function of Dombi operators. The proposed approximation formula is very simple as it has only one constant parameter. It does not include any exponential term, but has a good approximation accuracy and fulfills certain requirements that only a few of the known approximation formulas do.

**Keywords:** continuous logic, Dombi operators, sigmoid function, normal probability distribution, approximation

## 1 Introduction

The normal probability distribution plays a significant role in probability theory and mathematical statistics. Owing to the central limit theorem, it has an extremely wide range of applications in many areas of sciences. The fact that the

[a]Department of Computer Algorithms and Artificial Intelligence, University of Szeged, Árpád tér 2, H-6720 Szeged, Hungary, E-mail: dombi@inf.u-szeged.hu

[b]Institute of Business Economics, Eötvös Loránd University, Szép utca 2., H-1053 Budapest, Hungary, E-mail: jonas@gti.elte.hu

DOI: 10.14232/actacyb.23.3.2018.7

cumulative distribution function of the standard normal random variable cannot be expressed in a closed form and the practical needs for computing its values provided the motivations for researchers and practitioners over the last seven decades to approximate the standard normal probability distribution function. These research efforts resulted in an extremely wide range of approximations with many applications.

In this paper, we will introduce approximations to the standard normal probability distribution function that are connected with the well-known Dombi operators in continuous-valued logic. Firstly, we will utilize the averaging Dombi conjunction operator to construct a probability density function from two Sigmoid functions. We will show that this approximation method results in a probability distribution function that is identical with Tocher's approximation from 1963 [29]. Secondly, we will introduce the Kappa function and based on this function, we will construct the Quasi Logistic probability distribution function. We will show that the asymptotic Kappa function is just the Sigmoid function and using this result, we will also show how the Quasi Logistic probability distribution function can be utilized for approximating the standard normal probability distribution function. Here, we will point out how the proposed probability distribution function is connected with the generator function of Dombi operators and with the Kappa function-based uniary operator that can be utilized as a general fuzzy modifier operator. The novelty of our methods lies in the fact that some mathematical constructions of continuous-valued logic can be successfully utilized to construct approximations to probability distribution functions.

Many known approximations to the standard probability distribution function focus mainly on the approximation accuracy, and so these methods result in highly accurate functions, without taking some other aspects of the approximation into account. It should be mentioned here that we require our approximations to meet expectations that are based on certain theoretical and practical considerations. These expectations are simplicity and accuracy, asymptotic equality of the approximator function to the standard normal distribution function to first order at zero, symmetry of probability density function and a direct connection between the density and distribution functions.

Finally, we propose the use of the following probability distribution function, which is a special case of the Quasi Logistic distribution function, to approximate the standard normal probability distribution function:

$$\Phi_{\kappa,\pi}(x) = \begin{cases} 0, & \text{if } x \leq -\pi \\ \dfrac{1}{1+\left(\frac{\pi-x}{\pi+x}\right)^{\sqrt{2\pi}}}, & \text{if } x \in (-\pi,+\pi) \\ 1, & \text{if } x \geq +\pi. \end{cases} \tag{1}$$

We call the function $\Phi_{\kappa,\pi}(x)$ the Dombi-Jónás probability distribution function. It has only one constant parameter, which is the number $\pi$, while its maximum absolute approximation error over the set of real numbers is $2.36 \cdot 10^{-3}$. Note that

there are just a few known approximations with a single constant parameter in this accuracy range (e.g. [26], [20], [1], [13]), and all these approximations include exponential terms, while ours does not contain any and has a very simple form. It should also be added that the probability density function $\phi_{\kappa,\pi}(x)$ can be directly expressed in the terms of the probability distribution function $\Phi_{\kappa,\pi}(x)$ without differentiating it.

In many practical applications, the value of the standard normal probability distribution function for an argument being less than -3 or greater than +3 is considered to be zero and one, respectively, although the probability distribution does not take these values. The proposed $\Phi_{\kappa,\pi}(x)$ approximation has the value of zero, if $x \leq -\pi$, and it has the value of 1, if $x \geq +\pi$, so the function $\Phi_{\kappa,\pi}(x)$ may be viewed as an alternative, with bounded domain, to the standard normal probability distribution function.

The remaining part of the paper is organized as follows. In Section 2, we will review some notable approximations to the standard normal probability distribution function. Next, in Section 3, we will set our approximation criteria and introduce novel approximation methods that are connected with the Dombi operators. Lastly, in Section 4, we will summarize our approximation results and draw some key conclusions about the proposed Quasi Logistic probability distribution function.

## 2 Approximations to the Standard Normal Probability Distribution Function

Now, we will give a short review of the techniques that are widely used for approximating the standard normal probability distribution function and enumerate some notable approximations that have been constructed in the last seven decades.

We will use the common notations $\phi(x)$ and $\Phi(x)$ for the probability density function and probability distribution function of the standard normal random variable, respectively. That is,

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{x^2}{2}}; \ \Phi(x) = \int_{-\infty}^{x} \phi(t) \mathrm{d}t. \tag{2}$$

The approximation methods available in the literature can be categorized into two main approach categories [21]. One category is the group of approximations that are based on numerical methods, while the other category contains methods that are founded on ad-hoc approximations.

The numerical methods are typically based on numerical integration techniques, various power series, expansions in Hermite or Chebyshev polynomials and continued fraction expansions (e.g. [6], [18], [22], [25], [7]). In general, these methods can yield a high-level approximation accuracy, but require complex computations.

The ad-hoc approximation methods typically utilize an a priori selected parametric function and apply various mathematical techniques to estimate the parameters in order minimize the approximation error. Matic et al. [21], Soranzo

and Epure [28] and Yerukala and Boiroju [32] gave comprehensive overviews of the approximation formulas in their papers. Here, without striving for completeness, we enumerate some notable approximation formulas and indicate their maximum absolute errors (MAE).

1. Pólya (1949) [26]: $\Phi(x) \approx \frac{1+\sqrt{1-\mathrm{e}^{-2x^2/\pi}}}{2}$; $MAE = 3.15 \cdot 10^{-3}$
2. Hart (1957) [15]: $\Phi(x) \approx \frac{1}{\sqrt{2\pi}} \frac{\mathrm{e}^{-2x^2/\pi}}{x+0.8\mathrm{e}^{-0.4x}}$; $MAE = 4.30 \cdot 10^{-3}$
3. Tocher (1963) [29]: $\Phi(x) \approx \frac{\mathrm{e}^{2\sqrt{2/\pi}}}{1+\mathrm{e}^{2\sqrt{2/\pi}}}$ ; $MAE = 1.77 \cdot 10^{-2}$
4. Zelen & Severo (1964) [34]: $\Phi(x) \approx 1 - \left(a_1 t - a_2 t^2 + a_3 t^3\right) \frac{\mathrm{e}^{-\frac{x^2}{2}}}{\sqrt{2\pi}}$, where $t = (1 + 0.33267x)^{-1}$, $a_1 = 0.4361836$, $a_2 = 0.1201676$, $a_3 = 0.937298$; $MAE = 1.15 \cdot 10^{-5}$
5. Hart (1966) [16]: $\Phi(x) \approx 1 - \frac{\mathrm{e}^{-\frac{x^2}{2}}}{\sqrt{2\pi}x} \left(1 - \frac{\frac{\sqrt{1+bx^2}}{1+ax^2}}{P_0 + \sqrt{P_0 x^2 + \mathrm{e}^{-\frac{x^2}{2}} \frac{\sqrt{1+bx^2}}{1+ax^2}}}\right)$, where $a = \frac{1+\sqrt{1-2\pi^2+6\pi}}{2\pi}$, $b = 2\pi a^2$ and $P_0 = \sqrt{\pi/2}$; $MAE = 5.23 \cdot 10^{-5}$
6. Page (1977) [24]: $\Phi(x) \approx \frac{1}{2} \frac{1}{1+\tanh(y)}$, where $y = \sqrt{\frac{2}{\pi}} x \left(1 + 0.044715x^2\right)$; $MAE = 1.79 \cdot 10^{-4}$
7. Hamaker (1978) [14]: $\Phi(x) \approx 1 - \frac{1}{2}\left(1 - \sqrt{1 - \mathrm{e}^{-y^2}}\right)$, where $y = 0.806x(1 - 0.018x)$; $MAE \approx 6.23 \cdot 10^{-4}$
8. Lin (1989) [19]: $\Phi(x) \approx 1 - \frac{1}{2}\mathrm{e}^{-0.717x - 0.416x^2}$; $MAE = 6.59 \cdot 10^{-3}$
9. Norton (1989) [23]: $\Phi(x) \approx \begin{cases} 1 - \frac{1}{2}\mathrm{e}^{-0.717x-0.416x^2}, & \text{if } 0 \le x \le 2.7 \\ \frac{1}{\sqrt{2\pi}x}\mathrm{e}^{-\frac{x^2}{2}}, & \text{if } x > 2.7; \end{cases}$ $MAE = 8.07 \cdot 10^{-3}$
10. Lin (1990) [20]: $\Phi(x) \approx 1 - \frac{1}{1+\mathrm{e}^{4.2\pi\frac{x}{9-x}}}$, where $0 \le x < 9$; $MAE = 6.69 \cdot 10^{-3}$
11. Bagby (1995) [2]: $\Phi(x) \approx \frac{1}{2} + \frac{1}{2}\sqrt{1 - \frac{1}{30}\left(7\mathrm{e}^{-x^2/2} + 16\mathrm{e}^{-x^2(2-\sqrt{2})} + \left(7 + \frac{\pi}{4}x^2\right)\mathrm{e}^{-x^2}\right)}$; $MAE \approx 3.00 \cdot 10^{-5}$
12. Waissi & Rossin (1996) [31]: $\Phi(x) \approx \frac{1}{1+\mathrm{e}^{-\sqrt{\pi}\left(0.9x + 0.0418198x^3 + 0.0004406x^5\right)}}$; $MAE = 4.37 \cdot 10^{-5}$
13. Bryc (2002) [4]: $\Phi(x) \approx \frac{x^2 + a_1 x + a_2}{\sqrt{2\pi}x^3 + b_1 x^2 + b_2 x + 2a_2} \mathrm{e}^{-\frac{x^2}{2}}$, where $a_1 = 5.575192695$, $a_2 = 12.77436324$, $b_1 = 14.38718147$, $b_2 = 31.53531977$; $MAE = 1.87 \cdot 10^{-5}$

14. Shore (2005) [27]: $\Phi(x) \approx \frac{1+g(-x)+g(x)}{2}$,
where $g(x) = \mathrm{e}^{-\log 2\mathrm{e}^{\alpha/(\lambda/S_1)((1+S_1x)^{(\lambda/S_1)}-1)+S_2x}}$
$\lambda = 0.61228883$, $S_1 = 0.11105481$, $S_2 = 0.44334159$, $\alpha = 6.37309208$;
$MAE \approx 10^{-7}$

15. Aludaat and Alodat (2008) [1]: $\Phi(x) \approx \frac{1}{2} + \frac{1}{2}\sqrt{1 - \mathrm{e}^{-\sqrt{\pi/8}x^2}}$;
$MAE = 1.97 \cdot 10^{-3}$

16. Bowling et al. (2009) [3]: $\Phi(x) \approx \frac{1}{1+\mathrm{e}^{-(0.07056x^3+1.5976x)}}$; $MAE = 1.4 \cdot 10^{-4}$

17. Yerukala et al. (2011) [33]:
$$\Phi(x) \approx \begin{cases} 0.5 - 1.136H_1 + 2.47H_2 - 3.013H_3, & \text{if } 0 \le x \le 3.36 \\ 1, & \text{if } x > 3.36, \end{cases}$$
where $H_1 = \tanh(-0.2695x)$, $H_2 = \tanh(0.5416x)$ and $H_3 = \tanh(0.4134x)$;
$MAE = 1.25 \cdot 10^{-3}$

18. Vazquez-Leal et al. (2012) [30]: $\Phi(x) \approx \frac{1}{2}\tanh\left(\frac{179x}{23} - \frac{11}{2}\arctan\left(\frac{37x}{294}\right)\right) + \frac{1}{2}$;
$MAE \approx 1.00 \cdot 10^{-6}$

19. Choudhury (2014) [5]: $\Phi(x) \approx 1 - \frac{1}{\sqrt{2\pi}} \frac{\mathrm{e}^{-\frac{x^2}{2}}}{0.226+0.64x+0.33\sqrt{x^2+3}}$;
$MAE = 1.93 \cdot 10^{-4}$

20. Yerukala & Boiroju (2015) [32]: $\Phi(x) \approx 1 - \frac{\mathrm{e}^{-\frac{x^2}{2}}}{\frac{44}{79}+\frac{8}{5}x+\frac{5}{6}\sqrt{x^2+3}}$;
$MAE = 1.10 \cdot 10^{-4}$

21. Yerukala & Boiroju (2015) [32]: $\Phi(x) \approx w\Phi_1(x) + (1-w)\Phi_2(x)$,
where $x > 0$, $w = 0.268$, $\Phi_1(x)$ is the approximation by Hart (1966)
and $\Phi_2(x)$ is the approximation by Bryc (2002); $MAE = 7.54 \cdot 10^{-6}$

22. Matic et al. (2016) [21]:
$\Phi(x) \approx \frac{1}{2} + \frac{\mathrm{sgn}(x)}{2}\sqrt{1 - \mathrm{e}^{-\frac{2x^2}{\pi}(1+\gamma_2x^2+\gamma_4x^4+\gamma_6x^6+\gamma_8x^8+\gamma_{10}x^{10})}}$,
where $\gamma_2 = -\frac{1}{3} + \frac{1}{\pi}$; $\gamma_4 = \frac{7}{90} - \frac{2}{3\pi} + \frac{4}{3\pi^2}$; $\gamma_6 = -\frac{1}{70} + \frac{4}{15\pi} - \frac{4}{3\pi^2} + \frac{2}{\pi^3}$;
$\gamma_8 = \frac{83}{37800} - \frac{76}{945\pi} + \frac{34}{45\pi^2} - \frac{8}{3\pi^3} + \frac{16}{5\pi^4}$;
$\gamma_{10} = -\frac{73}{249480} + \frac{283}{14175\pi} - \frac{178}{567\pi^2} + \frac{88}{45\pi^3} - \frac{16}{3\pi^4} + \frac{16}{3\pi^5}$;
$MAE = 5.79 \cdot 10^{-6}$

23. Eidous and Al-Salman (2016) [13]: $\Phi(x) \approx \frac{1}{2}\left(1 + \sqrt{\mathrm{e}^{-5/8x^2}}\right)$;
$MAE = 1.81 \cdot 10^{-3}$

Based on the above approximation formulas, we may state that the accuracy of approximations increases with the complexity of formulas and with the number of parameters they possess.

# 3 Novel Methods based on Operators of Continuous-valued Logic

Fist of all, we will lay down some expectations that we require from approximations and use these criteria to evaluate our results and compare them with some well-known ones. Next, we will introduce the Dombi operators that are familiar in continuous-valued logic and construct novel approximation methods that are connected with these operators.

## 3.1 Expectations towards Our Approximations

The most basic expectation towards an approximation is that it is sufficiently accurate. In the literature, there are many approximations to the standard probability distribution function that focus mainly on the approximation accuracy. These efforts have resulted in highly accurate functions, without taking some other features of the approximation into account. Here we set some criteria – driven by theoretical and practical considerations – that we require our approximations to meet.

**Simplicity and accuracy.** The approximation functions should have a simple, easily computable formula, and the approximation accuracy should meet the requirements of practical applications.

**Identity to first order at zero.** Let $F(x)$ be an approximating function to the standard normal probability distribution function. We require $F(x)$ to be a probability distribution function and meet the following criteria:

$$\begin{gathered} F(0) = \Phi(0) = 0.5 \\ \frac{\mathrm{d}F(x)}{\mathrm{d}x}\bigg|_{x=0} = \frac{\mathrm{d}\Phi(x)}{\mathrm{d}x}\bigg|_{x=0} = \phi(x)\bigg|_{x=0} = \frac{1}{\sqrt{2\pi}}. \end{gathered} \tag{3}$$

**Symmetry.** Since the probability density function $\phi(x)$ is an even function, $\Phi(-x) = 1 - \Phi(x)$ holds for any $x \in \mathbb{R}$. We require the approximation $F(x)$ to have the same feature; that is, $F(-x) = 1 - F(x)$ for any $x \in \mathbb{R}$. Note that if $F(x)$ satisfies the $F(-x) = 1 - F(x)$ requirement, then the approximation error function $\delta(x) = \Phi(x) - F(x)$ is an odd function, and so the curve of $|\Phi(x) - F(x)|$ is symmetric with respect to the vertical axis.

**Direct connection between the density and distribution functions.** In practice, it may be useful, if the probability distribution function can be expressed by the probability density function without integration, and vice versa, if the probability density function can be expressed by the probability distribution function without differentiation. Hence, we prefer the approximations that result in probability density and distribution functions with a direct connection between them; that is, one can be expressed by the other one in a closed form.

It is worth emphasizing that only a few of the known approximations listed in Section 2 meet all the requirements we demanded. In general, the more complex an approximation formula is, the less of our criteria it meets. However, the approximations with more complex formulas and higher number of constant parameters result in a higher approximation precision. Note that many of the known approximations work just with positive values of variable $x$ and let the user compute the approximating function value by using the $\Phi(-x) = 1 - \Phi(x)$ equation for negative values of $x$.

## 3.2 Dombi Operators in Continuous-valued Logic

Here, we will introduce the Dombi operator class that can be utilized for implementing the conjunction and disjunction operations in continuous-valued logic [8], [10].

**Definition 1.** *The Dombi conjunction and disjunction operator in continuous-valued logic is given by*

$$o_\alpha(\mathbf{x}) = \frac{1}{1 + \left( \sum\limits_{i=1}^{n} \left( \frac{1-x_i}{x_i} \right)^\alpha \right)^{1/\alpha}} \text{ and } \overline{o}_\alpha(\mathbf{x}) = \frac{1}{1 + \left( \frac{1}{n} \sum\limits_{i=1}^{n} \left( \frac{1-x_i}{x_i} \right)^\alpha \right)^{1/\alpha}}, \tag{4}$$

*where* $\mathbf{x} = (x_1, x_2, ..., x_n)$*, and* $x_1, x_2, ..., x_n$ *are continuous-valued logic variables.*

If $\alpha > 0$, then the Dombi operator is a conjunction operator; if $\alpha < 0$, then it is a disjunction operator. Here, we will use the Dombi conjunction operators with two operands and $\alpha = 1$:

$$c(x_1, x_2) = o(x_1, x_2)|_{\alpha=1} = \begin{cases} 0, & \text{if } x_1 = 0 \text{ or } x_2 = 0 \\ \dfrac{1}{1 + \frac{1-x_1}{x_1} + \frac{1-x_2}{x_2}}, & \text{otherwise,} \end{cases} \tag{5}$$

$$\overline{c}(x_1, x_2) = \overline{o}(x_1, x_2)|_{\alpha=1} = \begin{cases} 0, & \text{if } x_1 = 0 \text{ or } x_2 = 0 \\ \dfrac{1}{1 + \frac{1}{2}\left( \frac{1-x_1}{x_1} + \frac{1-x_2}{x_2} \right)}, & \text{otherwise,} \end{cases} \tag{6}$$

where $x_1$ and $x_2$ are two continuous-valued logic variables. We call $\overline{c}$ the averaging Dombi conjunction operator. Note that operation $c$ is not idempotent, while $\overline{c}$ may be viewed as an idempotent variant of $c$.

**Remark 1.** Based on the general representation theorem [9],

$$o(\mathbf{x}) = f^{-1}\left( \sum_{i=1}^{n} f(x_i) \right) \text{ and } \overline{o}(\mathbf{x}) = f^{-1}\left( \frac{1}{n} \sum_{i=1}^{n} f(x_i) \right) \tag{7}$$

are strict operators, if $f(x)$ is a strictly monotone function, where $\mathbf{x} = (x_1, x_2, ..., x_n)$, and $x_1, x_2, ..., x_n$ are continuous-valued logic variables. If we apply the function

$$f(x) = f_\alpha(x) = \left(\frac{1-x}{x}\right)^\alpha \tag{8}$$

to $o(\mathbf{x})$ and $\overline{o}(\mathbf{x})$, then we get the operators $o_\alpha(\mathbf{x})$ and $\overline{o}_\alpha(\mathbf{x})$, respectively. That is, $f_\alpha(x)$ is the generator function of Dombi conjunction and disjunction operators.

In fuzzy logic, the linguistic modifiers like "very", "more or less", "somewhat", "rather" and "quite" over fuzzy sets that have strictly monotonously increasing or decreasing membership functions can be modeled by the following unary operator called the Kappa function [11].

**Definition 2.** *The Kappa modifier operator (Kappa function) is given by*

$$\kappa^{(\lambda)}_{\nu,\nu_0}(x) = \frac{1}{1 + \frac{1-\nu_0}{\nu_0}\left(\frac{\nu}{1-\nu}\frac{1-x}{x}\right)^\lambda}, \tag{9}$$

*where $\nu, \nu_0 \in (0,1)$, $\lambda \in \mathbb{R}$, and $x$ is a continuous-valued logic variable.*

In Section 3.4, we will use a special form of the unary modifier operator in (9) to construct a probability distribution function.

## 3.3 The Sigmoid Function and Some of Its Basic Properties

Since we will use the Sigmoid function to construct probability density and probability distribution functions, here we will introduce it and some of its main properties.

**Definition 3.** *The Sigmoid function $\sigma^{(\lambda_\sigma)}(x)$ with the parameter $\lambda_\sigma$ is given by*

$$\sigma^{(\lambda_\sigma)}(x) = \frac{1}{1 + \mathrm{e}^{-\lambda_\sigma x}}, \tag{10}$$

*where $\lambda_\sigma \in \mathbb{R}$, $\lambda_\sigma \neq 0$, $x \in \mathbb{R}$.*

Note that the Sigmoid function is also known as the Logistic function. The main properties, such as the range, continuity, monotonicity, limits, role of the parameter and convexity of the Sigmoid function $\sigma^{(\lambda_\sigma)}(x)$ are as follows.

**Range.** The range of $\sigma^{(\lambda_\sigma)}(x)$ is the interval $(0,1)$.

**Continuity.** $\sigma^{(\lambda_\sigma)}(x)$ is a continuous function in $\mathbb{R}$.

**Monotonicity.**

- If $\lambda_\sigma > 0$, then $\sigma^{(\lambda_\sigma)}(x)$ is strictly monotonously increasing
- If $\lambda_\sigma < 0$, then $\sigma^{(\lambda_\sigma)}(x)$ is strictly monotonously decreasing

**Limits.** Function $\sigma^{(\lambda_\sigma)}(x)$ takes neither the value zero, nor the value 1, as these are its limits:

$$\lim_{x\to+\infty} \sigma^{(\lambda_\sigma)}(x) = \begin{cases} 1, & \text{if } \lambda_\sigma > 0 \\ 0, & \text{if } \lambda_\sigma < 0, \end{cases} \tag{11}$$

$$\lim_{x\to-\infty} \sigma^{(\lambda_\sigma)}(x) = \begin{cases} 1, & \text{if } \lambda_\sigma < 0 \\ 0, & \text{if } \lambda_\sigma > 0. \end{cases} \tag{12}$$

**Role of the parameter.** The parameter $\lambda_\sigma$ of $\sigma^{(\lambda_\sigma)}(x)$ has a semantic meaning related to the shape of the function curve. The first derivative of $\sigma^{(\lambda_\sigma)}(x)$ at $x = 0$ is

$$\left.\frac{\mathrm{d}\sigma^{(\lambda_\sigma)}(x)}{\mathrm{d}x}\right|_{x=0} = \lambda_\sigma \sigma^{(\lambda_\sigma)}(0)\left(1 - \sigma^{(\lambda_\sigma)}(0)\right) = \frac{\lambda_\sigma}{4}. \tag{13}$$

That is, the $\lambda_\sigma$ parameter determines the slope of $\sigma^{(\lambda_\sigma)}(x)$ at $x = 0$.

**Convexity.**

- $\sigma^{(\lambda_\sigma)}(x)$ has a single inflection point that is at $x = 0$
- If $\lambda_\sigma > 0$, then $\sigma^{(\lambda_\sigma)}(x)$ changes from concave to convex at $x = 0$
- If $\lambda_\sigma < 0$, then $\sigma^{(\lambda_\sigma)}(x)$ changes from convex to concave at $x = 0$

Figure 1 shows some examples of Sigmoid function plots.

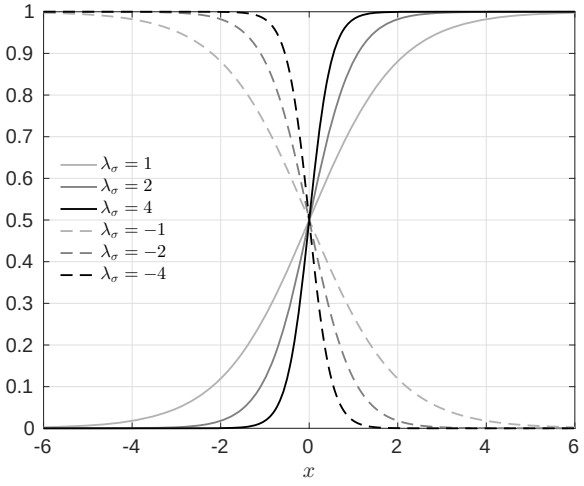



Figure 1: Examples of Sigmoid function plots.

### 3.4 Tocher's Approximation and the Averaging Dombi Conjunction Operator

Applying the averaging Dombi conjunction in (6) to $\sigma^{(\lambda_\sigma)}(x)$ and $\sigma^{(-\lambda_\sigma)}(x)$ yields the following $d_{\lambda_\sigma}(x)$ function:

$$\begin{aligned} d_{\lambda_\sigma}(x) = \overline{c}\left(\sigma^{(\lambda_\sigma)}(x), \sigma^{(-\lambda_\sigma)}(x)\right) = \frac{1}{1+\frac{1}{2}\left(\frac{1-\sigma^{(\lambda_\sigma)}(x)}{\sigma^{(\lambda_\sigma)}(x)}+\frac{1-\sigma^{(-\lambda_\sigma)}(x)}{\sigma^{(-\lambda_\sigma)}(x)}\right)} = \\ = \frac{1}{1+\frac{1}{2}\left(\mathrm{e}^{-\lambda_\sigma x}+\mathrm{e}^{\lambda_\sigma x}\right)} = \frac{2\mathrm{e}^{\lambda_\sigma x}}{\left(1+\mathrm{e}^{\lambda_\sigma x}\right)^2}. \end{aligned} \tag{14}$$

Figure 2 shows the averaging Dombi conjunction of two Sigmoid fuzzy membership functions; that is, the intersection of two fuzzy sets that are given by Sigmoid functions: by a decreasing and an increasing Sigmoid function with the same absolute $\lambda_\sigma$ parameter values.

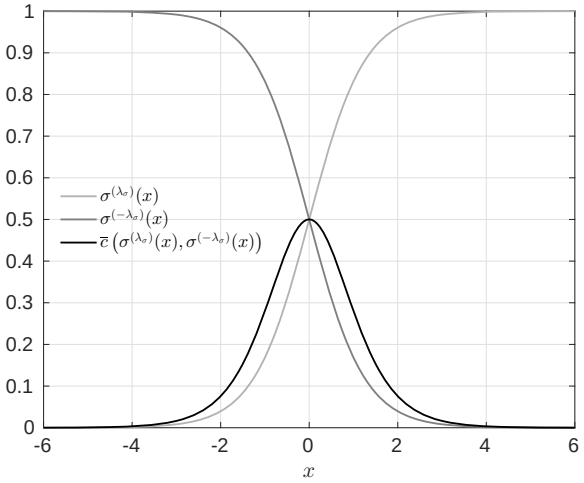



Figure 2: The averaging Dombi conjunction of two Sigmoid fuzzy membership functions.

Function $d_{\lambda_\sigma}(x)$, like the density function $\phi(x)$, has a bell-shaped curve, but since

$$\int_{-\infty}^{+\infty} d_{\lambda_\sigma}(x)\mathrm{d}x = \int_{-\infty}^{+\infty} \frac{2\mathrm{e}^{\lambda_\sigma x}}{\left(1+\mathrm{e}^{\lambda_\sigma x}\right)^2} = \left[-\frac{2}{\lambda_\sigma\left(1+\mathrm{e}^{\lambda_\sigma x}\right)}\right]_{-\infty}^{+\infty} = \frac{2}{\lambda_\sigma}, \tag{15}$$

$d_{\lambda_\sigma}(x)$ is not a probability density function. Hence,

$$\int_{-\infty}^{+\infty} \frac{\lambda_\sigma}{2} d_{\lambda_\sigma}(x)\mathrm{d}x = \int_{-\infty}^{+\infty} \frac{\lambda_\sigma \mathrm{e}^{\lambda_\sigma x}}{\left(1+\mathrm{e}^{\lambda_\sigma x}\right)^2} = 1, \tag{16}$$

and so we define the probability density function $\phi_\sigma(x)$ as follows.

**Definition 4.** *The probability density function $\phi_\sigma(x)$ is given by*

$$\phi_\sigma(x) = \frac{\lambda_\sigma \mathrm{e}^{\lambda_\sigma x}}{\left(1 + \mathrm{e}^{\lambda_\sigma x}\right)^2}, \tag{17}$$

*where $\lambda_\sigma = 2\sqrt{2/\pi}$.*

Note that setting $\lambda_\sigma$ to $2\sqrt{2/\pi}$ ensures that

$$\phi_\sigma(x)\big|_{x=0} = \phi(x)\big|_{x=0}. \tag{18}$$

The corresponding probability distribution function $\Phi_\sigma(x)$ is

$$\Phi_\sigma(x) = \int_{-\infty}^{x} \phi_\sigma(t)\mathrm{d}t = \left[-\frac{1}{1 + \mathrm{e}^{2\sqrt{2/\pi}t}}\right]_{-\infty}^{x} = \frac{1}{1 + \mathrm{e}^{-2\sqrt{2/\pi}x}}. \tag{19}$$

This means that the probability distribution function $\Phi_\sigma(x)$ is a Sigmoid function that has the parameter $\lambda_\sigma = 2\sqrt{2/\pi}$. It is worth adding here that $\Phi_\sigma(x)$ is identical to Tocher's approximation result in (3) from 1963 [29]. However, we derived the function $\Phi_\sigma(x)$ by generating the density function $\phi_\sigma(x)$ from Sigmoid functions by utilizing the averaging Dombi conjunction operator, and this approach is different from Tocher's.

**Approximation accuracy.** It can be shown numerically that

$$\max_{x \in \mathbb{R}} |\Phi(x) - \Phi_\sigma(x)| \approx 0.0177. \tag{20}$$

Figure 3 shows the curve of absolute error function $|\Phi(x) - \Phi_\sigma(x)|$.

**Properties of the approximation.** Here, we summarize the properties of this approximation in the light of expectations that were prescribed in Section 3.1.

- **Simplicity and accuracy.** $\Phi_\sigma(x)$ has a simple formula, but its maximum absolute approximation error has an order of magnitude of -2.
- **Identity to first order at zero.** Since $\Phi_\sigma(0) = \Phi(0)$ and the parameter $\lambda_\sigma$ of $\phi_\sigma(x)$ was set such that $\phi_\sigma(0) = \phi(0)$, $\Phi_\sigma(x)$ and $\Phi(x)$ are identical to first order at $x = 0$.
- **Symmetry.** The probability density function $\phi_\sigma(x)$ is an even function and so $\Phi_\sigma(-x) = 1 - \Phi_\sigma(x)$ holds for any $x \in \mathbb{R}$.
- **Direct connection between the density and distribution functions.** There is an interesting relation between the probability density function $\phi_\sigma(x)$

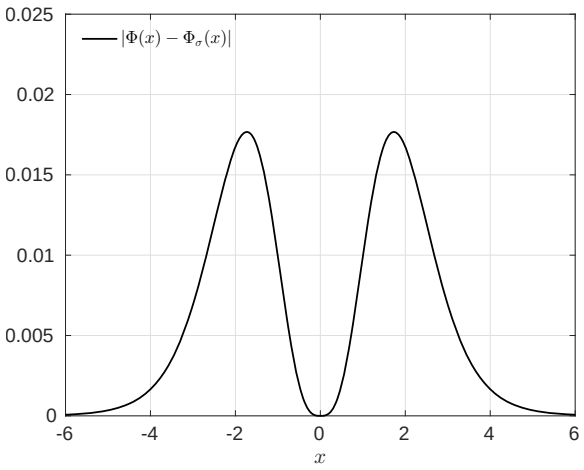



Figure 3: Absolute errors of approximation by $\Phi_\sigma(x)$.

and the probability distribution function $\Phi_\sigma(x)$ that is worth mentioning here. Namely, utilizing (17) and $\lambda_\sigma = 2\sqrt{2/\pi}$ gives the following equation:

$$\phi_\sigma(x) = 2\sqrt{\frac{2}{\pi}}\Phi_\sigma(x)\left(1 - \Phi_\sigma(x)\right). \tag{21}$$

That is, $\phi_\sigma(x)$ can be expressed in terms of $\Phi_\sigma(x)$ in a closed form.

According to Hillier and Liberman [17], the Sigmoid function that matches $\Phi(x)$ the best, has only one parameter and the form of (10) is

$$\Phi_{HL}(x) = \frac{1}{1 + \mathrm{e}^{-1.702x}}. \tag{22}$$

This approximation has a maximum absolute error of 0.0095. Note that although this approximation yields a higher accuracy than the approximation by the $\Phi_\sigma(x)$ function, the first derivative of $\Phi_{HL}(x)$ at $x = 0$ is not $1/\sqrt{2\pi}$; that is, $\Phi_{HL}(x)$ is not identical with $\Phi(x)$ to first order.

Note that if we used the Dombi conjunction operator in (5) to create a probability density function from two Sigmoid functions, then we would get the following probability distribution function:

$$\Phi^*_\sigma(x) = \frac{3}{\pi}\arctan\left(\frac{\sqrt{3}}{3}\left(2\mathrm{e}^{\sqrt{6\pi}/3x} + 1\right)\right) - \frac{1}{2}. \tag{23}$$

The maximum absolute error of this approximation is 0.0231 and calculation of the approximation formula requires the computation of an exponential function and an arcus tangent function. We can find simpler formulas with better precision values among the known approximations enumerated in Section 2.

## 3.5 An Approximation Connected with the Unary Modifier Operator

### 3.5.1 The Epsilon Function

Here, we introduce the Epsilon function that we will utilize for constructing approximations to the standard normal probability distribution function.

**Definition 5.** *The Epsilon function* $\varepsilon_d^{(\lambda)}(x)$ *is given by*

$$\varepsilon_d^{(\lambda)}(x) = \left(\frac{x+d}{d-x}\right)^{\lambda\frac{d}{2}}, \tag{24}$$

*where* $\lambda \in \mathbb{R}$, $\lambda \neq 0$, $d \in \mathbb{R}$, $d > 0$, $x \in (-d, +d)$.

The following theorem introduces an important asymptotic property of the Epsilon function.

**Theorem 1.** *For any* $x \in (-d, +d)$, *if* $d \to \infty$,

$$\varepsilon_d^{(\lambda)}(x) \to \mathrm{e}^{\lambda x}. \tag{25}$$

*Proof.* Let $x$ have a fixed value, $x \in (-d, +d)$.

$$\begin{aligned}\lim_{d\to\infty} \varepsilon_d^{(\lambda)}(x) = \lim_{d\to\infty} \left(\frac{x+d}{d-x}\right)^{\lambda\frac{d}{2}} = \lim_{d\to\infty} \left(\left(\frac{d-x+2x}{d-x}\right)^{d}\right)^{\frac{\lambda}{2}} = \\ = \lim_{d\to\infty} \left(\left(1+\frac{2x}{d-x}\right)^{d}\right)^{\frac{\lambda}{2}}.\end{aligned} \tag{26}$$

Since $x$ is fixed, if $d \to \infty$, then $\Delta = d - x \to \infty$ and so the previous equation can be continued as follows:

$$\begin{aligned}\lim_{d\to\infty} \left(\left(1+\frac{2x}{d-x}\right)^{d}\right)^{\frac{\lambda}{2}} = \lim_{\Delta\to\infty} \left(\left(1+\frac{2x}{\Delta}\right)^{\Delta+x}\right)^{\frac{\lambda}{2}} = \\ = \left(\lim_{\Delta\to\infty} \left(1+\frac{2x}{\Delta}\right)^{\Delta} \lim_{\Delta\to\infty} \left(1+\frac{2x}{\Delta}\right)^{x}\right)^{\frac{\lambda}{2}} = \left(\mathrm{e}^{2x}\right)^{\frac{\lambda}{2}} \cdot 1^{\frac{\lambda}{2}} = \mathrm{e}^{\lambda x}.\end{aligned} \tag{27}$$

□

Based on Theorem 1, we can state that the asymptotic Epsilon function is just the exponential function. It is worth mentioning here that the Epsilon function is the basis of the so-called Epsilon probability distribution, which can be utilized to approximate the exponential probability distribution [12].

### 3.5.2 The Kappa Function and Some of its Basic Properties

Here, we define the Kappa function that we will use to approximate the standard normal probability distribution function.

**Definition 6.** *The Kappa function* $\kappa_d^{(\lambda_\kappa)}(x)$ *is given by*

$$\kappa_d^{(\lambda_\kappa)}(x) = \frac{1}{1 + \left(\frac{d-x}{d+x}\right)^{\lambda_\kappa}}, \tag{28}$$

*where* $\lambda_\kappa \in \mathbb{R}$, $\lambda_\kappa > 0$, $d \in \mathbb{R}$, $d > 0$, $x \in (-d, +d)$.

Note that we utilize the Kappa function $\kappa_d^{(\lambda_\kappa)}(x)$ solely with positive $\lambda_\kappa$ parameter values. Here, we state the most important properties of the Kappa function $\kappa_d^{(\lambda_\kappa)}(x)$; namely, range, continuity, monotonicity, limits, role of the parameters and convexity.

**Range.** The range of $\kappa_d^{(\lambda_\kappa)}(x)$ is the interval $(0, 1]$.

**Continuity.** $\kappa_d^{(\lambda_\kappa)}(x)$ is a continuous function in $(-d, +d)$.

**Monotonicity.** As $\lambda_\kappa > 0$, $\kappa_d^{(\lambda_\kappa)}(x)$ is strictly monotonously increasing in the interval $(-d, +d)$.

**Limits.**

$$\lim_{x \to -d^+} \kappa_d^{(\lambda_\kappa)}(x) = 0 \tag{29}$$

$$\lim_{x \to +d^-} \kappa_d^{(\lambda_\kappa)}(x) = 1 \tag{30}$$

Note that as $\lambda_\kappa > 0$, $\kappa_d^{(\lambda_\kappa)}(x)$ takes the value of 1 at $d$.

**Role of the parameters.** Both parameters $\lambda_\kappa$ and $d$ of $\kappa_d^{(\lambda_\kappa)}(x)$ have a semantic meaning related to the shape of the function curve.

- Parameter $d$ specifies the $(-d, +d)$ domain of $\kappa_d^{(\lambda_\kappa)}(x)$.
- The first derivative of $\kappa_d^{(\lambda_\kappa)}(x)$ at $x = 0$ is

$$\left.\frac{\mathrm{d}\kappa_d^{(\lambda_\kappa)}(x)}{\mathrm{d}x}\right|_{x=0} = 2\lambda_\kappa d \left.\frac{\kappa_d^{(\lambda_\kappa)}(x)\left(1 - \kappa_d^{(\lambda_\kappa)}(x)\right)}{(d-x)(d+x)}\right|_{x=0} = \frac{\lambda_\kappa}{2d}. \tag{31}$$

That is, parameter $\lambda_\kappa$ determines the gradient of function $\kappa_d^{(\lambda_\kappa)}(x)$ at $x = 0$.

**Convexity.** It can be shown that the Kappa function $\kappa_d^{(\lambda_\kappa)}(x)$ has a single inflection point at $x = 0$, where it changes its shape from convex to concave.

### 3.5.3 Connection with the Sigmoid function

The Kappa function has the following asymptotic property that allows us to use it for approximating the Sigmoid function and through that the standard normal probability distribution function.

**Lemma 1.** *If $\sigma^{(\lambda_\sigma)}(x)$ is a Sigmoid function with the parameter $\lambda_\sigma > 0$, $\kappa_d^{(\lambda_\kappa)}(x)$ is a Kappa function with parameters $\lambda_\kappa$, $d > 0$ and*

$$\lambda_\kappa = \lambda_\sigma \frac{d}{2}, \tag{32}$$

*then for any $x \in (-d, +d)$, if $d \to \infty$, then*

$$\kappa_d^{(\lambda_\kappa)}(x) \to \sigma^{(\lambda_\sigma)}(x). \tag{33}$$

*Proof.* Let $x$ have a fixed value. If the conditions of the lemma are satisfied, then the Kappa function $\kappa_d^{(\lambda_\kappa)}(x)$ may be written as

$$\kappa_d^{(\lambda_\kappa)}(x) = \frac{1}{1 + \left(\frac{d-x}{d+x}\right)^{\lambda_\kappa}} = \frac{1}{1 + \left(\frac{d+x}{d-x}\right)^{-\lambda_\sigma \frac{d}{2}}} = \frac{1}{1 + \varepsilon_d^{(-\lambda_\sigma)}(x)}, \tag{34}$$

and based on Theorem 1, $\varepsilon_d^{(-\lambda_\sigma)}(x) \to \mathrm{e}^{-\lambda_\sigma x}$, if $d \to \infty$; that is,

$$\kappa_d^{(\lambda_\kappa)}(x) = \frac{1}{1 + \varepsilon_d^{(-\lambda_\sigma)}(x)} \xrightarrow[d\to\infty]{} \frac{1}{1 + \mathrm{e}^{-\lambda_\sigma x}} = \sigma^{(\lambda_\sigma)}(x). \tag{35}$$

□

**Corollary 1.** *In the interval $(-d, +d)$, the probability distribution function*

$$\Phi_\sigma(x) = \frac{1}{1 + \mathrm{e}^{-\lambda_\sigma x}} \tag{36}$$

*can be approximated by the Kappa function*

$$\kappa_d^{(\lambda_\kappa)}(x) = \frac{1}{1 + \left(\frac{d-x}{d+x}\right)^{\lambda_\kappa}}, \tag{37}$$

*where $d \in \mathbb{R}, d > 0$, $\lambda_\sigma = 2\sqrt{2/\pi}$, $\lambda_\kappa = \sqrt{2/\pi}d$.*

*Proof.* The corollary follows from Lemma 1. □

### 3.5.4 The Quasi Logistic Probability Distribution Function

Now, we will we define the Quasi Logistic probability distribution function by utilizing the Kappa function given in (28).

**Definition 7.** *The Quasi Logistic probability distribution function is given by*

$$\Phi_{\kappa,d}(x) = \begin{cases} 0, & \text{if } x \leq -d \\ \kappa_d^{(\lambda_\kappa)}(x), & \text{if } x \in (-d,+d) \\ 1, & \text{if } x \geq +d, \end{cases} \tag{38}$$

*where* $d \in \mathbb{R}, d > 0$, $\lambda_\kappa = \sqrt{2/\pi}d$.

It is worth mentioning here that there is an interesting relation between the Quasi Logistic probability density function $\phi_{\kappa,d}(x)$ and the probability distribution function $\Phi_{\kappa,d}(x)$.

**Lemma 2.** *If* $x \in (-d,+d)$, *then*

$$\phi_{\kappa,d}(x) = 2d^2\sqrt{\frac{2}{\pi}}\frac{\Phi_{\kappa,d}(x)\left(1-\Phi_{\kappa,d}(x)\right)}{(d-x)\,(d+x)}, \tag{39}$$

*where* $d \in \mathbb{R}, d > 0$.

*Proof.* Based on the definition of $\Phi_{\kappa,d}(x)$ in (38), if $x \in (-d,+d)$, then

$$\Phi_{\kappa,d}(x) = \kappa_d^{(\lambda_\kappa)}(x). \tag{40}$$

Utilizing this equation, (31) and $\lambda_\kappa = \sqrt{2/\pi}d$, we get

$$\begin{gathered} \phi_{\kappa,d}(x) = \frac{\mathrm{d}\Phi_{\kappa,d}(x)}{\mathrm{d}x} = \frac{\mathrm{d}\kappa_d^{(\lambda_\kappa)}(x)}{\mathrm{d}x} = \\ = 2\lambda_\kappa d\frac{\kappa_d^{(\lambda_\kappa)}(x)\left(1-\kappa_d^{(\lambda_\kappa)}(x)\right)}{(d-x)\,(d+x)} = 2d^2\sqrt{\frac{2}{\pi}}\frac{\Phi_{\kappa,d}(x)\left(1-\Phi_{\kappa,d}(x)\right)}{(d-x)\,(d+x)}. \end{gathered} \tag{41}$$

□

Utilizing Lemma 2, the Quasi Logistic probability density function $\phi_{\kappa,d}(x)$ for $x \in \mathbb{R}$ is

$$\phi_{\kappa,d}(x) = \begin{cases} 2d^2\sqrt{\frac{2}{\pi}}\frac{\kappa_d^{(\lambda_\kappa)}(x)\left(1-\kappa_d^{(\lambda_\kappa)}(x)\right)}{(d-x)\,(d+x)}, & \text{if } x \in (-d,+d) \\ 0, & \text{otherwise,} \end{cases} \tag{42}$$

where $d \in \mathbb{R}, d > 0$.

Note that based on the properties of $\kappa_d^{(\lambda_\kappa)}(x)$, it can be shown that $\Phi_{\kappa,d}(x)$ is in fact a probability distribution function and $\phi_{\kappa,d}(x)$ is its probability density function. Therefore, the following criteria are met:

1. $\phi_{\kappa,d}(x) \geq 0$ for any $x \in \mathbb{R}$
2. $\int_{-\infty}^{+\infty} \phi_{\kappa,d}(x)\mathrm{d}x = 1$
3. $\int_{-\infty}^{x} \phi_{\kappa,d}(t)\mathrm{d}t = \Phi_{\kappa,d}(x)$.

**Corollary 2.** *The standard normal probability distribution function $\Phi(x)$ can be approximated by the Quasi Logistic probability distribution function $\Phi_{\kappa,d}(x)$.*

*Proof.* The corollary follows from the fact that $\Phi(x)$ can be approximated by the Sigmoid function $\sigma^{(\lambda_\sigma)}(x)$ that has the parameter $\lambda_\sigma = 2\sqrt{2/\pi}$ and from Corollary 1 and from the definition of the Quasi Logistic probability distribution function. □

It is worth mentioning that $\phi_{\kappa,d}(x)$ can be derived from the Kappa function also in the following way. Utilizing the fact that

$$f_{\lambda_\kappa,d}(x) = \begin{cases} \dfrac{\mathrm{d}\kappa_d^{(\lambda_\kappa)}(x)}{\mathrm{d}x}, & \text{if } x \in (-d,+d) \\ 0, & \text{otherwise} \end{cases} \tag{43}$$

is a probability density function,

$$\frac{\mathrm{d}\kappa_d^{(\lambda_\kappa)}(x)}{\mathrm{d}x} = 2\lambda_\kappa d \frac{\kappa_d^{(\lambda_\kappa)}(x)\left(1-\kappa_d^{(\lambda_\kappa)}(x)\right)}{(d-x)\,(d+x)}, \tag{44}$$

and setting the requirement $f_{\lambda_\kappa,d}(0) = \phi(0)$ results in the following equation:

$$2\lambda_\kappa d \frac{\kappa_d^{(\lambda_\kappa)}(x)\left(1-\kappa_d^{(\lambda_\kappa)}(x)\right)}{(d-x)\,(d+x)}\bigg|_{x=0} = \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{x^2}{2}}\bigg|_{x=0}. \tag{45}$$

Using (31), this equation leads to $\lambda_\kappa = \sqrt{2/\pi}d$; that is, $f_{\lambda_\kappa,d}(x) = \phi_{\kappa,d}(x)$, if $\lambda_\kappa = \sqrt{2/\pi}d$.

### 3.5.5 Approximation Accuracy

It can be shown numerically that $|\Phi(x) - \Phi_{\kappa,d}(x)|$ is approximately minimal, if $d = 3.1152$. In this case, the maximum absolute approximation error is $2.15 \cdot 10^{-3}$. Considering the fact that 3.1152 is close to $\pi$, using $d = \pi$ instead of $d = 3.1152$ does not worsen significantly the approximation accuracy. If $d = \pi$, then the maximum absolute approximation error is $2.36 \cdot 10^{-3}$. Although the parameter $d$ with value of $d = 3.1152$ yields the least maximum absolute approximation error among the Quasi Logistic probability distribution functions, we propose the use of function

$\Phi_{\kappa,\pi}(x)$ as it has a very simple form and its maximum absolute approximation error is just slightly greater than that of function $\Phi_{\kappa,d}(x)$ with $d = 3.1152$.

$$\Phi_{\kappa,\pi}(x) = \begin{cases} 0, & \text{if } x \leq -\pi \\ \dfrac{1}{1+\left(\frac{\pi-x}{\pi+x}\right)^{\sqrt{2\pi}}}, & \text{if } x \in (-\pi, +\pi) \\ 1, & \text{if } x \geq +\pi \end{cases} \tag{46}$$

We call the Quasi Logistic probability distribution function with $d = \pi$; that is, the function $\Phi_{\kappa,\pi}(x)$, the Dombi-Jónás probability distribution function. The absolute errors $|\Phi(x) - \Phi_{\kappa,d}(x)|$ for $d = 3.1152$ and $d = \pi$ are shown in Figure 4.

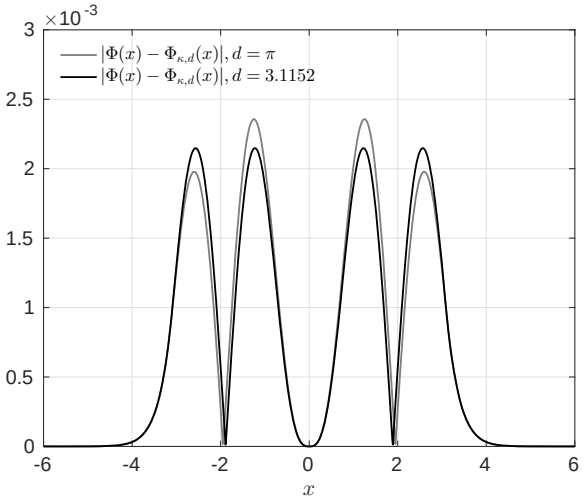



Figure 4: Absolute errors of approximations by using Quasi Logistic probability distribution functions.

### 3.5.6 Properties of the Approximation

Here, we summarize the properties of the $\Phi_{\kappa,\pi}(x)$ approximation.

- **Simplicity and accuracy.** The maximum absolute error of approximation $\Phi_{\kappa,\pi}(x)$ is $2.36 \cdot 10^{-3}$ and at the same time function $\Phi_{\kappa,\pi}(x)$ has a very simple form. In this accuracy range, there is no other known approximation that has such a simple form. The known approximations that yield higher accuracy have more complex forms, while the ones with similarly complex formulas do not give greater accuracy.
- **Identity to first order at zero.** Since $\Phi_{\kappa,\pi}(0) = \Phi(0)$ and the probability density function $\phi_{\kappa,\pi}(x)$ was constructed such that $\phi_{\kappa,\pi}(0) = \phi(0)$, $\Phi_{\kappa,\pi}(x)$ and $\Phi(x)$ are identical to first order at $x = 0$.

- **Symmetric absolute error function.** It can be shown that the probability density function $\phi_{\kappa,\pi}(x)$ is an even function and so $\Phi_{\kappa,\pi}(-x) = 1 - \Phi_{\kappa,\pi}(x)$ holds for any $x \in \mathbb{R}$.
- **Direct connection between the density and distribution functions.** Based on Lemma 2, the density function $\phi_{\kappa,\pi}(x)$ can be directly expressed in terms of the distribution function $\Phi_{\kappa,\pi}(x)$ in a closed form.

### 3.5.7 Connections with Dombi Operators

Next, we will show how the Epsilon function $\varepsilon_d^{(-\lambda)}(x)$ and the Kappa function $\kappa_d^{(\lambda_\kappa)}(x)$ are connected with the Dombi operators.

**Lemma 3.** *The generator function $f_\alpha(x)$ of Dombi conjunction and disjunction operators can be derived from the Epsilon function $\varepsilon_d^{(-\lambda)}(x)$ by a linear function transformation.*

*Proof.* Let us apply the $x' = (x+d)/(2d)$ linear transformation to the variable $x$, where $x \in (-d, d)$, $d > 0$. After this transformation, the domain of $x'$ is the interval $(0,1)$, $x = 2dx' - d$, and

$$\begin{aligned}\varepsilon_d^{(-\lambda)}(x) = \left(\frac{x+d}{d-x}\right)^{-\lambda\frac{d}{2}} = \left(\frac{2dx'-d+d}{d-2dx'+d}\right)^{-\lambda\frac{d}{2}} = \left(\frac{x'}{1-x'}\right)^{-\lambda\frac{d}{2}} = \\ = \left(\frac{1-x'}{x'}\right)^{\lambda\frac{d}{2}} = f_\alpha(x'),\end{aligned} \tag{47}$$

where $\alpha = \lambda d/2$. □

Based on this result, the generator function of the Dombi operators may be viewed as a special case of the Epsilon function.

**Lemma 4.** *If $\nu = \nu_0 = 1/2$, then the Kappa function $\kappa_d^{(\lambda_\kappa)}(x)$ can be derived from the Kappa function $\kappa_{\nu,\nu_0}^{(\lambda)}(x)$ in (9) by applying a linear function transformation.*

*Proof.* The lemma can be proven by setting $\lambda = \lambda_\kappa$ and applying the $x' = 2dx - d$ linear transformation ($d > 0$). □

Based on this lemma, we can state that the Kappa function $\kappa_d^{(\lambda_\kappa)}(x)$, which we utilized to construct the Quasi Logistic probability distribution function, is a special case of the general fuzzy modifier operators.

## 4 Conclusions and Future Work

Table 1 summarizes the maximum absolute errors of the approximations presented earlier. From this table, we can see that the approximation by function $\Phi_{\kappa,\pi}(x)$ has a one order of magnitude less maximum absolute error than the approximation by

Table 1: Goodness of the approximations given previously

| $F(x)$ | $\max_{x\in\mathbb{R}} \lvert\Phi(x) - F(x)\rvert$ |
|---|---|
| $\Phi_\sigma(x)$ | $1.77 \cdot 10^{-2}$ |
| $\Phi_{\kappa,\pi}(x)$ | $2.36 \cdot 10^{-3}$ |

the function $\Phi_\sigma(x)$. Figure 5 and Figure 6 show the approximating function curves and the absolute errors of the approximations, respectively.

Based on comparisons of these approximations with the ones given in the literature, the following findings should be emphasized.

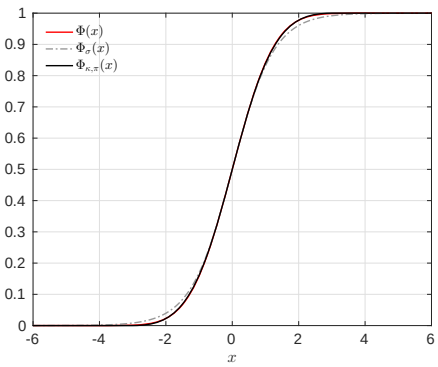



Figure 5: Approximations.

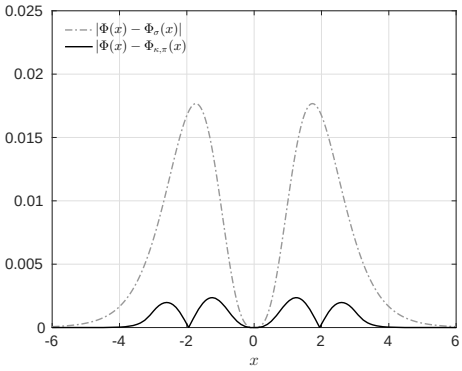



Figure 6: Absolute errors.

**Simplicity and accuracy.** The first of the approximations listed earlier, which is the same as Tocher's approximation [29], has a maximum absolute approximation error of $1.77 \cdot 10^{-2}$. Although this approximation has a simple form, its accuracy is lower than the accuracy of some known approximations that have similar complex formulas (e.g. [26], [20], [1], [13]). The maximum absolute error of approximation by function $\Phi_{\kappa,\pi}(x)$ is $2.36 \cdot 10^{-3}$. This error is one order of magnitude less than that of the first approximation. At the same time, function $\Phi_{\kappa,\pi}(x)$ has a very simple formula with only one constant parameter which is the constant $\pi$. It should be added here that there are only a few known approximations with a single constant parameter in this accuracy range (e.g. [26], [20], [1], [13]), and all these approximations include exponential terms, while $\Phi_{\kappa,\pi}(x)$ does not contain any. That is, to the best of our knowledge, in this accuracy range, there is no other known approximation that has such a simple formula as $\Phi_{\kappa,\pi}(x)$. The known approximations that yield a higher accuracy have more complex formulas, while the ones with similar complex formulas do not give a higher accuracy.

**Identity to first order at zero.** The presented approximations of $\Phi(x)$ are identical with $\Phi(x)$ to first order at $x = 0$.

**Symmetric absolute error function.** It is worth noting that both of the above approximations meet the $F(-x) = 1 - F(x)$ criterion for any $x \in \mathbb{R}$, and so their absolute error function curves are symmetric with respect to the vertical axis, as can be seen in Figure 6.

**Direct connection between the density and distribution functions.** It is the case both for the Sigmoid approximation $\Phi_\sigma(x)$ and the Quasi Logistic approximation $\Phi_{\kappa,d}(x)$ that the probability density function can be directly expressed in terms of the probability distribution function in a closed form. That is, the density function can be derived from the distribution function without differentiating it. This property of the of our approximations can be very useful in practice.

**Connections with the possibilistic approach.** The given approximators are connected with continuous logic. Namely, the approximation $\Phi_\sigma(x)$ is derived from Sigmoid fuzzy membership functions by applying the averaging Dombi conjunction operator, while the Quasi Logistic approximation is a linearly transformed form of the Kappa function that is a well-known modifier operator in fuzzy theory.

**Applicability.** For any $x \in \mathbb{R}$ argument, the standard normal probability distribution function $\Phi(x)$ takes a value in the interval $(0, 1)$. In other words, it associates positive probabilities with arguments that are much less than 0, and gives probabilities less than 1 for those arguments that are much greater than zero. In many practical applications, the probabilities for arguments that are much less or much greater than the expected value of a normally distributed random variable are considered to be zero and one, respectively, although the exact probabilities for these arguments lie in the interval $(0, 1)$. The probability distribution function $\Phi_{\kappa,\pi}(x)$ takes a value from the interval $(0, 1)$ only if its argument is greater than $-\pi$ and less than $+\pi$. Noting that $\Phi(-\pi) = 0.00084$, $\Phi(\pi) = 0.99916$ and

$$\max_{x\in(-\pi,+\pi)} |\Phi(x) - \Phi_{\kappa,\pi}(x)| \approx 2.36 \cdot 10^{-3}, \tag{48}$$

the Dombi-Jónás probability distribution may be viewed as an alternative, with bounded domain, to the standard normal probability distribution.

**Plans for future work.** The Kappa function that we used to construct the probability distribution function $\Phi_{\kappa,\pi}(x)$ is symmetric about the point $(0, 0.5)$. In certain economic and technological applications, asymmetric probability distributions with bounded domains are needed for modeling and simulation purposes. As part of our future research work, we would like to study how a generalized, asymmetric version of the Kappa function, which is defined over the bounded domain $(a, b)$, can be utilized for constructing asymmetric probability distribution functions.

# References

[1] Aludaat, K. M. and Alodat, M. T. A note on approximating the normal distribution function. *Applied Mathematical Sciences*, 2(9):425–429, 2008.

[2] Bagby, R. J. Calculating normal probabilities. *The American Mathematical Monthly*, 102:46–49, 1995.

[3] Bowling, S., Khasawneh, M., Kaewkuekool, S., and Cho, B. A logistic approximation to the cumulative normal distribution. *Journal of Industrial Engineering and Management*, 2(1):114–127, 2009.

[4] Bryc, W. A uniform approximation to the right normal tail integral. *Applied Mathematics and Computation*, 127(2-3):365–374, 2002.

[5] Choudhury, A. A simple approximation to the area under standard normal curve. *Mathematics and Statistics*, 2(3):147–149, 2014.

[6] Cuyt, A. A. M., Petersen, V., Verdonk, B., Waadeland, H., and Jones, W. B. *Handbook of Continued Fractions for Special Functions*. Springer, Berlin/New York, 2008.

[7] Divgi, D. R. Calculation of univariate and bivariate normal probability functions. *The Annals of Statistics*, 7(4):903–910, 1979.

[8] Dombi, J. A general class of fuzzy operators, the DeMorgan class of fuzzy operators and fuzziness measures induced by fuzzy operators. *Fuzzy Sets and Systems*, 8(2):149 – 163, 1982.

[9] Dombi, J. Properties of the fuzzy connectives in the light of the general representations theorem. *Acta Cybernetica*, 7(3):313–321, 1986.

[10] Dombi, J. Towards a general class of operators for fuzzy systems. *IEEE Transactions on Fuzzy Systems*, 16(2):477–484, 2008.

[11] Dombi, J. On a certain type of unary operators. In *2012 IEEE International Conference on Fuzzy Systems*, pages 1–7, June 2012.

[12] Dombi, J., Jónás, T., and Tóth, Zs. E. The epsilon probability distribution and its application in reliability theory. *Acta Polytechnica Hungarica*, 15(1):197–216, 2018.

[13] Eidous, O. and Al-Salman, S. One-term approximation for normal distribution function. *Mathematics and Statistics*, 4(1):15–18, 2016.

[14] Hamaker, H.C. Approximating the cumulative normal function and its inverse. *Applied Statistics*, 27:76–77, 1978.

[15] Hart, R. G. A formula for the approximation of definite integrals of the normal distribution function. *Mathematicl Tables and ther Aids to Computation*, 11(60):265–265, October 1957.

[16] Hart, R. G. A close approximation related to the error function. *Mathematics of Computation*, 20(96):600–602, 1966.

[17] Hillier, F. S. and Lieberman, G. J. *Introduction to Operations Research, 7th Ed.* McGraw-Hill, New York, USA, 2001.

[18] Lee, Chu-In Charles. On laplace continued fraction for the normal integral. *Annals of the Institute of Statistical Mathematics*, 44(1):107–120, 1992.

[19] Lin, Jinn-Tyan. Approximating the normal tail probability and its inverse for use on a pocket calculator. *Applied Statistics*, 38:69–70, 1989.

[20] Lin, Jinn-Tyan. A simpler logistic approximation to the normal tail probability and its inverse. *Applied Statistics*, 39:255–257, 1990.

[21] Matic, I., Radoicic, R., and Stefanica, D. A sharp Polya-based approximation to the normal cdf. *SSRN*, 2016. http://dx.doi.org/10.2139/ssrn.2842681.

[22] Moran, P. A. P. Calculation of the normal distribution function. *Biometrika*, 67:675–676, 1980.

[23] Norton, R. M. Pocket-calculator approximation for areas under the standard normal curve. *The American Statistician*, 43:24–26, 1989.

[24] Page, E. Approximations to the cumulative normal function and its inverse for use on a pocket calculator. *Applied Statistics*, 26:75–76, 1977.

[25] Patel, J. K. and Read, C. B. *Handbook of the normal distribution.* Marcel Dekker Inc, 1996.

[26] Pólya, G. Remarks on computing the probability integral in one and two dimensions. In *Proceedings of the 1st Berkeley Symposium on Mathematical Statistics and Probability*, pages 63–78, Berkeley, Calif., 1949. University of California Press.

[27] Shore, Ha. Accurate RMM-based approximations for the CDF of the normal distribution. *Communications in Statistics - Theory and Methods*, 34(3):507–513, 2005.

[28] Soranzo, A. and Epure, E. Very simply explicitly invertible approximations of normal cumulative and normal quantile function. *Applied Mathematical Sciences*, 8(87):4323–4341, 2014.

[29] Tocher, K. D. *The Art of Simulation.* English University Press, London, 1963.

[30] Vazquez-Leal, H., Castaneda-Sheissa, R., Filobello-Nino, U., Sarmiento-Reyes, A., and Orea, J. Sanchez. High accurate simple approximation of normal distribution integral. *Mathematical Problems in Engineering*, 2012(ID:124029), 2012.

[31] Waissi, Gary R. and Rossin, Donald F. A sigmoid approximation of the standard normal integral. *Applied Mathematics and Computation*, 77:91–95, 1996.

[32] Yerukala, R. and Boiroju, N. K. Approximating to the cumulative normal function and its inverse. *International Journal of Scientific & Engineering Research*, 6(4):515–518, 2015.

[33] Yerukala, R., Boiroju, N. K., and Reddy, M. K. An approximation to the cdf of standard normal distribution. *International Journal of Mathematical Archive*, 2(7):1077–1079, 2011.

[34] Zelen, M. and Severo, N. C. Probability functions. In Abramowitz, M. and Stegun, I. A., editors, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Applied Mathematics Series, pages 925–995. Washington:National Bureau of Standards, 1964.

*Received 25th February 2018*

Acta Cybernetica **23** (2018) 853–865.

# The Convergence Time for Selfish Bin Packing*

György Dósa[a] and Leah Epstein[b]

**Abstract**

In classic bin packing, the objective is to partition a set of $n$ items with positive rational sizes in $(0, 1]$ into a minimum number of subsets called bins, such that the total size of the items of each bin at most 1. We study a bin packing game where the cost of each bin is 1, and given a valid packing of the items, each item has a cost associated with it, such that the items that are packed into a bin share its cost equally. We find tight bounds on the exact worst-case number of steps in processes of convergence to pure Nash equilibria. Those are processes that are given an arbitrary packing as an initial packing. As long as there exists an item that can reduce its cost by moving from its bin to another bin, in each step, a controller selects such an item and instructs it to perform such a beneficial move. The process converges when no further beneficial moves exist. The tight function of $n$ that we find is in $\Theta(n^{3/2})$. This improves the previous bound of Ma et al. [14], who showed an upper bound of $O(n^2)$.

## 1 Introduction

We study a class of bin packing games, that are based on the well-known standard bin packing problem [17, 4, 6, 5], a basic combinatorial optimization problem. In this problem, a set of $n$ items $I = \{1, 2, \ldots, n\}$ is given, where the size of item $t$, denoted by $s_t$, satisfies $0 < s_t \leq 1$. The goal is to partition (or pack) the items into a minimum number of subsets or blocks. Each such block is *packed* into a unit capacity bin, and the load of a bin is defined to be the total size of items packed into it (and can never exceed 1). Here, we study bin packing from the point of view of algorithmic game theory.

We now define the game theoretical concepts required for the definition of the bin packing game. In a *strategic game*, there is a finite set of players, and a finite and non-empty set of strategies (or actions) that players can perform. Each player

*Supported by VKSZ_12-1-2013-0088 "Development of cloud based smart IT solutions by IBM Hungary in cooperation with the University of Pannonia" and by National Research, Development and Innovation Office – NKFIH under the grant SNN 116095.

[a]Department of Mathematics, University of Pannonia, Veszprém, Hungary, E-mail: dosagy@almos.vein.hu

[b]Department of Mathematics, University of Haifa, Haifa, Israel. E-mail: lea@math.haifa.ac.il

DOI: 10.14232/actacyb.23.3.2018.8

has to choose a strategy (possibly independently from other players). Each player has a cost for each one of the possible situations or outcomes, where an outcome is a possible set of strategies of all players, containing one strategy for each player. A classic form of a stable solution is a *Nash equilibrium* (NE) [21]. This is a kind of solution concept of a game with at least two players, where no player can decrease its cost by changing only its own strategy unilaterally. That is, if each player has chosen a (pure or mixed) strategy and no player can benefit by changing its strategy while the other players keep theirs unchanged, then the current set of strategy choices and the corresponding costs result in an outcome or solution that is a Nash equilibrium (NE). We are interested in pure Nash equilibria, where the actions of players are chosen in a deterministic way, and will discuss only this kind of NE.

Given an input for bin packing, the set of players are the items. The pure strategy of a player is the index of the bin into which it is packed (the number of possible bins is $n$, as this number of bins is always sufficient). We say that a bin $B \subseteq I$ is a valid bin if $\sum_{t \in B} s_t \leq 1$, that is, if its load does not exceed 1. Changing the strategy of an item means that it moves to be packed in a different (non-empty or empty) bin. For $0 \leq k \leq n$, we define a $k$-bin to be a bin that has exactly $k$ items, and a $k^+$-bin is a bin that has at least $k$ items. The cost of an item packed into a valid $k$-bin (for $k \geq 1$) is $\frac{1}{k}$. We let the cost of an item that is not packed into a valid bin be infinite. The deviation of an item $t$ packed in a $k_1$-bin $B_1$ (where $t$ is included in the number of items of $B_1$) to a $k_2$-bin $B_2$ (where $t$ is not included in the number of items of $B_2$) is beneficial if $s(B_2) + s_t \leq 1$ (since otherwise the cost of the item in the alternative bin is infinite) and $k_2 \geq k_1$ (as otherwise its cost is not reduced by moving). The standard bin packing problem can be therefore seen as a class of games, where every input corresponds to a game. The uniform cost sharing rule is motivated by the well-known egalitarian or Shapley model in game theory, which was introduced in an algorithmic game-theoretic context by Anshelevich et al. [2]. Unlike other bin packing games, the variant with uniform sharing is a congestion game.

In this paper, we are interested in convergence processes. Such a process receives a set of items and a packing. The packing obviously corresponds to an outcome of the game whose players are those items. The process stops when it reaches a solution that is an NE. As long as it is not an NE, a step is performed. In each step, a controller selects an item that can benefit (reduce its cost) by moving to another bin, and instructs it to move from its current bin to a specified bin (where its cost will be smaller). In each step a single item moves and decreases its cost, while other items may be affected (those that were packed with the moving item will have larger costs, and those that were packed into the bin where it moved will have smaller costs). It is shown by Ma et al. [14] (who were the first to study the variant with equal sharing of the bin costs) that such a process always converges in $O(n^2)$ steps. This kind of games are in fact singleton congestion games [15, 16], but the number of resources has an exponential size in the number of players, and it is not given explicitly (these are all possible subsets of items that can be packed into a bin), so the convergence and existence of NE can be deduced from previous

work on congestion games, but the polynomial time convergence cannot be deduced from those.

Bin packing in general, and more specifically bin packing games, have a number of applications [3, 9, 10]. Equal sharing is the simplest form of sharing, and it does not require prior information given by the players (who may or may not be truthful). Bin packing games where items share the cost of the bin proportionally (according to sizes) rather than equally was introduced by Bilò [3], who was the first to study the bin packing problem from this kind of game-theoretic perspective. He proved that every game in this class has an NE. He also proved that any such bin packing game converges to an NE after a finite (but possibly exponentially long) sequence of steps, starting from any initial configuration of the items. The time of convergence for this type of cost sharing was also studied in [19, 20]. Multiple papers studied the quality of NE and other types of equilibria [3, 9, 10, 7, 1]. Polynomial time algorithms that compute an NE for games with proportional cost sharing and for equal cost sharing are given in [22, 14, 7]. Note that the term "bin packing games" is used in the literature for a completely different type of games [12, 13, 18], and there is recent interest in those games as well.

**Our result.** We show that the worst-case number of steps for convergence is $\Theta(n^{1.5})$. The exact function expressing the worst-case number of steps is

$$\frac{i(i+1)(i-1)}{3} + j - ij \ ,$$

where

$$n = \frac{i(i+1)}{2} - j \quad \text{for} \quad 0 \leq j \leq i-1 \ .$$

We prove the lower bound by defining a sequence of steps, while the upper bound is proved using two potential functions, one of which is used in [14] and the other one is completely different. Interestingly, combining the two potential functions allows us to find a tight bound for any $n \geq 1$.

The maximal number of steps is achieved by a process that is not completely intuitive. The process is started with a packing where every item is packed into its own bin, and it ends with a packing where all items are packed into one bin. Moreover, it is obtained using items whose sizes are sufficiently small, such that all items can indeed be packed together. Those last properties of the initial packing and the final packing are natural as converting an arbitrary process to such a process, by reducing the sizes and adding steps in the beginning and at the end, obviously increases the number of steps. We show that the process of convergence can be split into two clear parts. In the first part, items migrate until a packing with a special structure which we call *a staircase packing* will be achieved. In such a packing, any non-empty bin contains different number of items, and these numbers (of items) are as small as possible (for example, if there are 17 items, there are five bins, containing one, two, three, five, and six items, respectively). After the staircase structure is built, it is destroyed again, to obtain fuller bins along time. The steps are carefully chosen such that the process is applied exactly in this way, and other steps are not performed. As we prove an exact bound (and not only an

order of growth), the process is defined very carefully. An important part of the proof of the upper bound is Lemma 6, where we consider an interesting invariant holding until a staircase packing is created.

A preliminary version of this work appeared as [8].

## 2 The exact convergence time

In [14], processes of the following kind were studied. The process starts with an arbitrary packing, and in each step one item that can reduce its cost by moving to another bin is selected by a controller and is moved to another bin such that its cost becomes smaller. The number of steps for convergence was shown to be $O(n^2)$ [14]. In this section we find the exact worst case number of steps, which turns out to be $\Theta(n^{3/2})$. Note that [19] showed using methods from [11] (where convergence for scheduling problems is studied) that for the case of proportional cost sharing, the number of steps can be exponential.

Given an integer $n \geq 1$, we let

$$i = \min\{h | h(h+1)/2 \geq n\} \quad \text{and} \quad j = i(i+1)/2 - n \ .$$

Thus, $n = i(i+1)/2 - j$, where $i \geq 1$, and $0 \leq j \leq i-1$ (since $n > i(i-1)/2 = i(i+1)/2 - i$). Additionally, since $n \leq i(i+1)/2 < (i+1)^2$ and $n > i(i-1)/2 > (i-1)^2/4$, we have $i > \sqrt{n} - 1$ and $i < 2\sqrt{n} + 1$, and thus $i = \Theta(\sqrt{n})$. We show that the maximum number of steps that can be performed for any set of items and initial configuration is exactly

$$\nabla_{i,j} = \frac{i(i+1)(i-1)}{3} + j - ij \ .$$

Note, that in case $i \geq 12$, the next inequalities are valid: $i - 1 \geq i/2$, $i/6 - 1 \geq i/12$, and $i + 1 \leq 2i$. Thus, it can be verified that

$$\nabla_{i,j} \leq \frac{i(i+1)(i-1)}{3} < i^3$$

and

$$\nabla_{i,j} > \frac{i^3}{6} - i^2 = i^2(i/6 - 1) \geq i^3/12 \ .$$

Thus, $\nabla_{i,j} = \Theta(n^{\frac{3}{2}})$.

We start with the lower bound.

**Lemma 1.** *For every positive integer $n$, there exists an input of $n$ items, for which there is an initial packing of these items, and a sequence of $\nabla_{i,j}$ steps that are performed until no additional steps can be done.*

*Proof.* Consider a set of $n$ items, each of size $\frac{1}{n}$, and an initial packing where each one of the items is packed in its own bin. Let a *staircase packing* be a packing where for every $1 \leq \eta \leq i$, $\eta \neq j$, there is exactly one bin with $\eta$ items.

We show using induction on $i$ that there exists a sequence of exactly $\Delta_{i,j} = i(i+1)(i-1)/6 - j(j-1)/2$ steps that results in a staircase packing. Note that for $i = 1$, it holds that $j = 0$, and thus $\Delta_{1,0} = 0$. Otherwise $i \geq 2$, and in this case $j \geq 0$ and $0 \leq j(j-1) \leq (i-1)(i-2)$ hold. Thus we get the next chain of inequalities:

$$i(i+1)(i-1) - 3j(j-1) \geq i(i+1)(i-1) - 3(i-1)(i-2) = (i-1)(i^2 - 2i + 6) > 0$$

where we used the property $i^2 - 2i + 6 \geq 6$. First, we show the claim for the case $j = 0$ (where $\Delta_{i,0} = i(i+1)(i-1)/6$) by induction on $i$.

For $i = 1$, in every packing there is exactly one bin with one item, and this packing is a staircase packing. For a given value of $i$, $n = i(i+1)/2$. We consider a subset of $n' = n - i = i(i-1)/2$ items. By the induction hypothesis it is possible to obtain a packing such that for any $1 \leq \eta \leq i-1$ there is a bin with $\eta$ items. Considering the complete set of $n$ items, we get that for any $2 \leq \eta \leq i-1$ there is a bin with $\eta$ items, and additionally there are $i+1$ bins, each with a single item. By the induction hypothesis, this packing is obtained in $i(i-1)(i-2)/6$ steps. Let $B_\eta$ denote a specific bin with $\eta$ items for $1 \leq \eta \leq i-1$, where the bin $B_1$ is chosen arbitrarily. The $i$ other items packed in dedicated bins are called *free items*. For $k = 1, \ldots, i-1$, the $k$-th free item is moved from its bin, to the bins $B_1$, $B_2$, …, $B_{i-k}$, in this order. $B_{i-k}$ will now contain $i-k+1$ items and will not be used again in this process. After all these steps, $B_\eta$ (for $1 \leq \eta \leq i-1$) will contain $\eta+1$ items. The $i$-th free item remains packed in its own bin, so as a result, for any $1 \leq \eta \leq i$ there is a bin with $\eta$ items. The number of additional steps for the free items (the steps that are applied after the bins $B_\eta$ are created using the induction hypothesis) is

$$\sum_{k=1}^{i-1} (i-k) = i(i-1)/2 \ ,$$

as the number of steps for the $k$th free item is $i-k$. The total number of steps is therefore

$$\frac{i(i-1)(i-2)}{6} + \frac{i(i-1)}{2} = \frac{i(i+1)(i-1)}{6} \ .$$

To show the claim for the case for $j \neq 0$ (and $i \geq 2$), we use the claim that was proved for $j = 0$. Assume that $n = i(i+1)/2 - j$ where $0 < j < i$. In this case, first we create a staircase packing of a subset of $n' = i(i-1)/2$ items, leaving $i-j$ free items. For $k = 1, \ldots, i-j$, the $k$-th free item is moved from its bin, to the bins $B_1$, $B_2$, …, $B_{i-k}$, in this order. The bin $B_{i-k}$ will contain $i-k+1$ items as a result and will not be used for later steps. After this is done for $i-j$ items, $B_\eta$ will contain $\eta+1$ items for $j \leq \eta \leq i-1$, and for $1 \leq \eta \leq j-1$, $B_\eta$ still contains $\eta$ items. Thus, for every $1 \leq \eta \leq i$, $\eta \neq j$, there is exactly one bin with $\eta$ items and this is exactly a staircase packing as required. The number of additional steps

(after the bins $B_\eta$ are created using the claim for $j = 0$) is

$$\sum_{k=1}^{i-j}(i-k) = i(i-1)/2 - j(j-1)/2 \ .$$

The total number of steps is

$$\frac{i(i+1)(i-1)}{6} - \frac{j(j-1)}{2} \ .$$

Once a staircase packing is achieved, we show that it is possible to reach a packing where all items are packed in one bin together using exactly $i(i+1)(i-1)/6 - ij + j(j+1)/2$ steps. We define a phase as follows. In the beginning of a phase there are bins with different numbers of items. Let

$$J = \{j_1 < j_2 < \cdots < j_{|J|}\}$$

be the set of numbers of items before some phase, and let the bin $B_\eta$ for $\eta \in J$ be the bin with $\eta$ items. If $|J| > 1$, we repeatedly take an item from $B_{j_1}$, and move it to $B_{j_2}$ then to $B_{j_3}$ and so forth until it reaches $B_{j_{|J|}}$. A phase ends when all items of $B_{j_1}$ were moved. If $j = 0$, then initially $J = \{1, \ldots, i\}$, there are $i-1$ phases, and the number of steps in all phases is

$$\sum_{k=1}^{i-1} k(i-k) = \frac{i^2(i-1)}{2} - \frac{(i-1)i(2i-1)}{6} = \frac{i(i-1)(i+1)}{6} \ .$$

Otherwise, initially $J = \{1, \ldots, i\} - \{j\}$, there are $i-2$ phases, and the number of steps is

$$\sum_{k=1}^{j-1} k(i-1-k) + \sum_{k=j+1}^{i-1} k(i-k) = \sum_{k=1}^{i-1} k(i-k) - \sum_{k=1}^{j-1} k - j(i-j)$$

$$= \frac{i^2(i-1)}{2} - \frac{i(i-1)(2i-1)}{6} - \frac{j(j-1)}{2} - j(i-j) = \frac{i(i+1)(i-1)}{6} - ij + \frac{j}{2} + \frac{j^2}{2} \ .$$

The total number of steps is therefore

$$\frac{i(i+1)(i-1)}{6} - \frac{j(j-1)}{2} + \frac{i(i-1)(i+1)}{6} + \frac{j}{2} - ij + \frac{j^2}{2} = \frac{i(i+1)(i-1)}{3} + j - ij = \nabla_{i,j} \ .$$

□

Next, we prove the main result of this paper.

**Theorem 1.** *The number of steps until convergence is at most*

$$\nabla_{i,j} = \frac{i(i+1)(i-1)}{3} + j - ij = \Theta(n^{\frac{3}{2}}) \ ,$$

*and there exists an input of $n$ items where this bound can be achieved.*

*Proof.* The lower bound was proved in the previous lemma. For the upper bound, consider an input $I$ of $n = i(i+1)/2 - j$ items for $0 \leq j \leq i-1$, an initial configuration and a sequence of moves. Let $p_{\min}$ denote the smallest item size in $I$. Let $\varepsilon = \min\{p_{\min}, 1/n\}$, and let $I'$ be the input where $s_t = \varepsilon$ for $1 \leq t \leq n$. For the input $I'$ there cannot be invalid moves, since all items can be packed into one bin.

**Lemma 2.** *The initial configuration and the sequence of moves of $I$ are valid for $I'$ as well.*

*Proof.* Since no item size was increased, all configurations of $I$ are valid for $I'$. Since the cost of an item in a packing depends only on numbers of items in its bin and not on their sizes, modifying the sizes may only increase sets of beneficial deviations, that is, every move that was beneficial and possible for $I$ remains such for $I'$ and the sequence of moves is still valid. □

In what follows, we will consider only sequences of moves for $I'$. In particular, we consider only sequences with a maximum number of moves. Such a sequence must exist since from the results of [14] every sequence of moves has a finite length.

**Lemma 3.** *Every sequence with a maximum number of moves starts with the configuration where every item is packed in a separate bin, and ends with the configuration that all items are packed in one bin.*

*Proof.* Consider a sequence of $\ell$ moves. Assume that there is a bin $B$ with $k \geq 2$ items in the initial configuration, and let $\phi \in B$. Modify the configuration such that instead of $B$ the starting configuration has the two bins $B \setminus \{\phi\}$ and $\{\phi\}$ (other bins remain unchanged). Next, add a step in the beginning of the sequence of moves where $\phi$ moves to join the items of $B \setminus \{\phi\}$. This is an improving step since $\phi$ reduces its cost from 1 to $\frac{1}{k}$. This results in a sequence of $\ell+1$ steps, which contradicts maximality.

Next, assume that after the sequence of moves there are at least two non-empty bins, containing $k_1$ and $k_2$ items respectively, where $k_1 \leq k_2$. Let $\psi$ be an item packed in the first bin. Add a move of $\psi$ to the second bin in the end of the sequence. This is an improving step since $\psi$ reduces its cost from $\frac{1}{k_1}$ to $\frac{1}{k_2+1} \leq \frac{1}{k_1+1} < \frac{1}{k_1}$. This results in a sequence of $\ell+1$ steps, which contradicts maximality. □

Let $k > 0$ be an integer. We define a *level $k$ small step* to be a move where an item moves from a $k$-bin to another $k$-bin. A step is called a *small step* if there is an integer $k$ such that the step is a level $k$ small step. Given the set of sequences of steps of maximum length we focus on sequences where the maximum length prefix of small steps has maximum length.

**Lemma 4.** *Assume that after a prefix of the sequence of steps is applied there are at least two $k$-bins. Then the first step in the remainder of the sequence of steps involving a $k$-bin is a level $k$ small step.*

*Proof.* Assume by contradiction that there is no level $k$ small step in the remaining part of the sequence. Since the sequence of steps terminates only when all items are packed in one bin, there is at least one item in the union of the $k$-bins that will perform a move (in fact, all the items of all the $k$-bins except for possibly one such bin will do that). Consider the first step after the current configuration was reached that involves a $k$-bin (either an item moving to the bin or moving out of it).

There are two possible moves. If an item $\psi$ moves from a $k$-bin into a bin with $k' > k$ items, we modify the sequence as follows. First $\psi$ moves to another $k$-bin, and then it moves to the bin with $k'$ items. The second step is still beneficial for $\psi$ since in the second step it moves from a $(k+1)$-bin to a bin with $k' \geq k+1$ items. This modification augments the length of the sequence by 1, which contradicts maximality.

If an item $\phi$ moves from a bin with $\tilde{k} < k$ items to one of the $k$-bins, we modify the sequence as follows. First choose an arbitrary item from one of the $k$-bins and move it to another $k$-bin. Then, move $\phi$ to the bin out of which the item was just moved (which now has $k-1$ items). This last move is beneficial since $\tilde{k} \leq k-1$. This modification augments the length of the sequence by 1, which contradicts maximality. □

**Lemma 5.** *Consider a maximum length prefix of small steps. After this prefix is performed, every bin has a different number of items.*

*Proof.* Assume by contradiction that at this time there are two $k$-bins. Using Lemma 4, there will be a level $k$ small step later in the sequence, which will be the first move that involves $k$-bins. Since all items are identical, it is possible to perform such a step immediately instead of at a later time. This does not change the number of steps in the sequence, and it increases the length of the maximum length prefix of small steps, which contradicts maximality of the prefix (out of sequences of maximum length). □

**Lemma 6.** *Consider the maximum length prefix of small steps. After this prefix is performed, there is one bin of each number of items in* $\{1, 2, \ldots, i\} \setminus \{j\}$*, that is, a staircase packing is created.*

*Proof.* We prove an invariant that is kept as long as only small steps are done. Let $b_k$ be the number of bins with $k$ items, and recall that initially $b_1 = n$ and $b_\ell = 0$ for $0 < \ell \leq n$. Assume that at a given time, $k_m$ is the maximum integer such that $b_{k_m} > 0$. We say that a number $1 \leq k \leq k_m - 1$ is bad if $b_k = 0$, and otherwise it is good. That is, a number $k$ is bad if there are no $k$-bins, but there exists at least one $(k+1)^+$-bin. If $b_k \geq 2$ then we say that $k$ is *very good*. Two bad numbers are called *consecutive* bad numbers if all numbers between them are good, that is, if $k_1$ and $k_2$ such that $k_1 < k_2 < k_m$ are both bad ($b_{k_1} = 0$ and $b_{k_2} = 0$), and for all $k'$ such that $k_1 < k' < k_2$, $b_{k'} > 0$.

The invariant is as follows. For every pair of consecutive bad numbers $k_1$, $k_2$, where $1 \leq k_1 < k_2 < k_m$, there exists a number $\tilde{k}$, where $k_1 < \tilde{k} < k_2$, such that $\tilde{k}$ is very good.

Initially, $k_m = 1$, thus there are no bad numbers, and the invariant holds trivially. Recall that we only analyze small steps and consider the change resulting from a single level $k$ small step. Every level $k$ small step implies that before this step there are at least two $k$-bins and so $k$ is very good.

Note that $k$ is the only number that can become bad as a result of a level $k$ small step. Moreover, if $k = k_m$, then the value $k_m$ increases by 1. Assume first that $k$ remains very good. No bad numbers are created, and since no number stops beings very good then the invariant holds (even if some number stops being bad). If $k$ remains good, but not very good, then still no new bad numbers are created and we only need to consider the case that $k$ was the only very good number between two consecutive bad numbers. Let these two numbers be $k_1 < k < k_2$. If $k_2 > k+1$ and $k_1 < k-1$, then the numbers of $k_1$-bins and $k_2$-bins are unchanged (that is, these numbers remain zero) and the numbers $k_1$, $k_2$ remain consecutive bad numbers between which we need to show that a very good number exists after the step. Since $k+1$ was good, as a result of the move $b_{k+1} \geq 2$, and since $k_1 < k+1 < k_2$, there is a very good number between $k_1$ and $k_2$, as required. If $k_1 = k-1$ but $k_2 > k+1$ then $k_1$ becomes good. If $k_1$ was the minimum bad number then we are done. Otherwise, let $k_3 < k_1$ be a bad number such that $k_3$ and $k_1$ were consecutive bad numbers. We now have that $k_3$ and $k_2$ are consecutive bad numbers and $b_{k+1} \geq 2$ so $k+1$ is a very good number between them. If $k_1 < k-1$ but $k_2 = k+1$ then $k_2$ becomes good. If $k_2$ was the maximum bad number then we are done. Otherwise, let $k_4 > k_2$ be a bad number such that $k_2$ and $k_4$ were consecutive bad numbers. We now have that $k_3$ and $k_4$ are consecutive bad numbers and $b_{k-1} \geq 2$ so $k-1$ is a very good number between them. Finally, if both $k_1 = k-1$ and $k_2 = k+1$ hold, then the only case of interest is when $k_1$ was not the minimum bad number and $k_2$ was not the maximum bad number. We let $k_3 < k_1$ be a bad number such that $k_3$ and $k_1$ were consecutive bad numbers, and let $k_4 > k_2$ be a bad number such that $k_4$ and $k_2$ were consecutive bad numbers. Now $k_3$ and $k_4$ are consecutive bad numbers. There is a very good number in $(k_3, k_1)$ which is now a very good number between $k_3$ and $k_4$.

Finally, we consider the case where $k$ becomes bad. If there previously was a bad number $k_2$ such that $k_2 > k$, we distinguish two cases. If $k_2 > k+1$, then $k$ and $k_2$ becomes a consecutive bad pair of numbers, and $k+1$ becomes a very good number between them. Otherwise, $k_2 = k+1$ becomes good. If $k_2$ was the maximum bad number then we are done, and otherwise, let $k_4 > k_2$ be such that $k_2$ and $k_4$ were consecutive bad numbers. Instead, $k$ and $k_4$ are now consecutive bad numbers, and the very good number between them is the same one which was very good between $k_2$ and $k_4$. The proof is symmetric for the case that there previously was a bad number $k_1$ such that $k_1 < k$.

To complete the proof, consider the configuration after the (maximum length) prefix of small steps. Since every bin has a different number of items, there are no very good numbers, and hence, by the invariant, there is at most one bad number.

If there exists a bin with at least $i+1$ items, and there is just one bad number, then there are at least $(i+1)(i+2)/2 - i = i(i+1)/2 + 1 > n$ items. If there is no bin with at least $i$ items, then there are at most $i(i-1)/2 < n$ items. Thus, there is a bin with $i$ items, and since there is at most one bad number, the bad number must be $j$ if $j \neq 0$, and otherwise there is no bad number. Therefore, the packing at this time is a staircase packing. □

**Lemma 7.** *The number of steps in the (maximum length) prefix of small steps is at most*

$$\frac{i(i+1)(i-1)}{6} - \frac{j(j-1)}{2} .$$

*Proof.* We use the potential function as in [14] which is the sum of squares of number of items in the bins. In the beginning every item is in a dedicated bin, so the potential is equal to $n = i(i+1)/2 - j$. Consider a level $k$ small step. The potential function increases by exactly 2 in this step, since the only change is that instead of two $k$-bins, there is a $(k-1)$-bin a $(k+1)$-bin, and the increase in the potential is exactly

$$(k+1)^2 + (k-1)^2 - 2k^2 = 2 .$$

Since a staircase packing is achieved in the (maximum length) prefix of small steps, the value of the potential after this prefix is

$$\sum_{k=1}^{i} k^2 - j^2 = \frac{i(i+1)(2i+1)}{6} - j^2 .$$

Thus, the number of steps cannot exceed half the difference between the final potential and the initial potential, which is

$$\left( \frac{i(i+1)(2i+1)}{6} - j^2 - \left( \frac{i(i+1)}{2} - j \right) \right) /2 = \frac{i(i+1)(i-1)}{6} - \frac{j(j-1)}{2} .$$

□

**Lemma 8.** *The number of steps in the remainder of the sequence after the (maximum length) prefix of small steps is at most*

$$\frac{i(i+1)(i-1)}{6} - ij + \frac{j(j+1)}{2} .$$

*Proof.* In this case we define a different potential function. Sort the bins in non-increasing order according to numbers of items. Let the index of an item be the index of the bin into which it is packed. The potential of a packing is sum of indices of items.

The potential is clearly positive at all times. The final potential is $n$, since all items are packed in one bin. Consider a step in which an item moves from a $k_1$-bin $B_v$ to a $k_2$-bin $B_u$ (where $k_2 \geq k_1$). Since all items are identical, we assume that $B_v$ is the $k_1$-bin of maximum index, and $B_u$ is the $k_2$ bin of minimum index. This

holds even if $k_1 = k_2$, since in this case there are at least two bins with this number of items. Since the bins are sorted by non-increasing order according to numbers of items we have $v > u$. As a result of the move, $B_v$ now has $k_1 - 1$ items, and $B_u$ now has $k_2 + 1$ items. By definition, if $u > 1$ then $B_{u-1}$ has at least $k_2 + 1$ items. Similarly, if $B_{v+1}$ exists then it has at most $k_1 - 1$ items, so the sorted order is still valid. The change in the potential in this step is the change in the index of the bin of the moving item, which is $v - u \geq 1$.

If $j = 0$, then the potential before the remainder of the sequence of moves is performed is

$$\sum_{k=1}^{i} k(i-k+1) = \frac{i(i+1)^2}{2} - \frac{i(i+1)(2i+1)}{6} = \frac{i(i+1)(i+2)}{6}$$

while $n = \frac{i(i+1)}{2}$, so the number of steps is at most

$$\frac{i(i+1)(i+2)}{6} - \frac{i(i+1)}{2} = \frac{i(i+1)(i-1)}{6} .$$

If $j > 0$, then the potential before the remainder of the sequence of moves is performed is

$$\sum_{k=1}^{i-j} k(i-k+1) + \sum_{k=i-j+1}^{i-1} k(i-k) = i\sum_{k=1}^{i-1} k + \sum_{k=1}^{i-j} k - \sum_{k=1}^{i-1} k^2$$

$$= \frac{i^2(i-1)}{2} + \frac{(i-j)(i-j+1)}{2} - \frac{i(i-1)(2i-1)}{6}$$

$$= \frac{i(i-1)(i+1)}{6} + \frac{i^2}{2} + \frac{j^2}{2} - ij + \frac{i}{2} - \frac{j}{2} .$$

In each step the function decreases by at least 1, so the number of steps is at most

$$\frac{i(i-1)(i+1)}{6} + \frac{i^2}{2} + \frac{j^2}{2} - ij + \frac{i}{2} - \frac{j}{2} - \frac{i(i+1)}{2} + j = \frac{i(i+1)(i-1)}{6} - ij + \frac{j}{2} + \frac{j^2}{2} .$$

□

Taking the sum of the maximum number of steps in the prefix (till the last small step of the maximum length prefix of small steps) with that of the remainder we get

$$\frac{i(i+1)(i-1)}{6} - \frac{j(j-1)}{2} + \frac{i(i+1)(i-1)}{6} - ij + \frac{j(j+1)}{2} = \frac{i(i+1)(i-1)}{3} + j - ij = \nabla_{i,j} .$$

□

## References

[1] R. Adar and L. Epstein. Selfish bin packing with cardinality constraints. *Theoretical Computer Science*, 495:66–80, 2013.

[2] E. Anshelevich, A. Dasgupta, J. M. Kleinberg, É. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM Journal on Computing*, 38(4):1602–1623, 2008.

[3] V. Bilò. On the packing of selfish items. In *Proc. of the 20th International Parallel and Distributed Processing Symposium (IPDPS'06)*. IEEE, 2006. 9 pages.

[4] E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: A survey. In D. Hochbaum, editor, *Approximation algorithms*. PWS Publishing Company, 1997.

[5] E. Coffman Jr. and J. Csirik. Performance guarantees for one-dimensional bin packing. In T. F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*, chapter 32, pages (32–1)–(32–18). Chapman & Hall/Crc, 2007.

[6] J. Csirik and G. J. Woeginger. On-line packing and covering problems. In *A. Fiat and G. J. Woeginger, editors,* Online Algorithms: The State of the Art, pages 147–177, 1998.

[7] Gy. Dósa and L. Epstein. Generalized selfish bin packing. CoRR, abs/1202.4080, 2012.

[8] Gy. Dósa and L. Epstein. The convergence time for selfish bin packing. In *Proc. of The 7th International Symposium on Algorithmic Game Theory (SAGT'14)*, 37–48, 2014.

[9] L. Epstein and E. Kleiman. Selfish bin packing. *Algorithmica*, 60(2):368–394, 2011.

[10] L. Epstein, E. Kleiman, and J. Mestre. Parametric packing of selfish items and the Subset Sum algorithm. *Algorithmica*, 74(1): 177–207, 2016.

[11] E. Even-Dar, A. Kesselman, and Y. Mansour. Convergence time to Nash equilibrium in load balancing. *ACM Transactions on Algorithms*, 3(3):32, 2007.

[12] U. Faigle and W. Kern. On some approximately balanced combinatorial cooperative games. *Mathematical Methods of Operations Research* 38(2), 141-152, 1993.

[13] U. Faigle and W. Kern Approximate core allocation for binpacking games. *SIAM Journal on Discrete Mathematics*, 11(3):387–399, 1998.

[14] R. Ma, Gy. Dósa, X. Han, H.-F. Ting, D. Ye, and Y. Zhang. A note on a selfish bin packing problem. *Journal of Global Optimization*, 56(4), 1457–1462, 2013.

[15] R. Holzman and N. Law-Yone. Strong equilibrium in congestion games. *Games and Economic Behavior*, 21(1-2):85101, 1997.

[16] S. Ieong, B. McGrew, E. Nudelman, Y. Shoham, and Q. Sun. Fast and compact: A simple class of congestion games, In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 489–494, 2005.

[17] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3:256–278, 1974.

[18] W. Kern and X. Qiu. Integrality gap analysis for bin packing games. *Operations Research Letters*, 40(5):360–363, 2012.

[19] F. K. Miyazawa and A. L. Vignatti. Convergence time to Nash equilibrium in selfish bin packing. *Electronic Notes in Discrete Mathematics*, 35:151–156, 2009.

[20] F. K. Miyazawa and A. L. Vignatti. Bounds on the convergence time of distributed selfish bin packing. *International Journal of Foundations of Computer Science*, 22(3):565–582, 2011.

[21] J. Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

[22] G. Yu and G. Zhang. Bin packing of selfish items. In *The 4th International Workshop on Internet and Network Economics (WINE'08)*, pages 446–453, 2008.

*Received 1st February 2018*

Acta Cybernetica **23** (2018) 867–886.

# Weighted Languages Recognizable by Weighted Tree Automata*

Zoltán Fülöp[a] and Zsolt Gazdag[a]

**Abstract**

Yields of recognizable weighted tree languages, yields of local weighted tree languages, and weighted context-free languages are related. It is shown that the following five classes of weighted languages are the same: (i) the class of weighted languages generated by plain weighted context-free grammars, (ii) the class of weighted languages recognized by plain weighted tree automata, (iii) the class of weighted languages recognized by deterministic and plain top-down weighted tree automata, (iv) the class of weighted languages recognized by deterministic and plain bottom-up weighted tree automata, and (v) the class of weighted languages determined by plain weighted local systems.

## 1 Introduction

A tree automaton recognizes a set of trees over a ranked alphabet $\Sigma$ and a yield alphabet (or frontier alphabet) $X$ [14, 15]. Such trees are called $\Sigma X$-trees and the elements of $X$ may be leaves of $\Sigma X$-trees. Hence, a tree automaton also recognizes a language over $X$ as follows. For a $\Sigma X$-tree $\xi$, we define the yield $\mathrm{yd}(\xi)$ of $\xi$ to be the string in $X^*$ obtained by reading the leaves of $\xi$ from left to right. Then, the language recognized by a tree automaton is the set of all strings $\mathrm{yd}(\xi)$, where $\xi$ is a tree recognized by the automaton.

The idea of using tree automata in the theory of languages was proposed already in papers [26], [20], [27] and [22]. Then, more results were obtained in [7], [23], [28], and [25], of which a summary can be found in [14, 15] (also, cf. [10, 6]). Among other things, it was proved that the following four classes of languages are the same: (i) the class of context-free languages, (ii) the class of languages recognized by tree automata, (iii) the class of languages recognized by deterministic top-down tree automata, and (iv) the class of languages obtained by taking the yield of local tree languages (cf. Thm. II.9.4, III.2.7, and III. 2.9 in [14]).

With another line of research, tree automata were generalized to weighted tree automata (wta for short) [2, 1], in order to be able to deal with quantitative aspects

*This research was supported by the Hungarian Scientific Research Fund (OTKA) Grant K 108448.

[a]Department of Foundations of Computer Science, University of Szeged, Árpád tér 2, 6720 Szeged, Hungary, E-mail: {fulop,gazdag}@inf.u-szeged.hu

DOI: 10.14232/actacyb.23.3.2018.9

of recognizable tree languages. A wta recognizes a weighted $\Sigma X$-tree language; that is, a mapping from the set of $\Sigma X$-trees to a weight structure. Here, we consider the case that the weight structure is a semiring $K$. For surveys, see [11, 13]; and note that in these papers weighted tree languages are called tree series. Also, weighted context-free languages were introduced under the name of algebraic power series [5]; see [24, 19] and [21] for summary and [8] for a recent application[1].

Weighted $\Sigma X$-tree languages with a yield alphabet and weighted languages over $X$ may be related as in the classical (unweighted) case. We can generalize the yield function to the weighted setting such that the yield $\mathrm{yd}(\Phi)$ of a weighted $\Sigma X$-tree language $\Phi$ will be a weighted language over $X$. In fact, the weight of a string $w \in X^*$ in $\mathrm{yd}(\Phi)$ is the sum of the weight of all trees in $\Phi$ of which the yield is $w$. We note that there may be infinitely many such trees, hence the sum may have infinitely many terms. In this case the semiring $K$ should be complete in the sense defined in [9].

The fundamental relation between recognizable weighted tree languages and weighted context-free languages is established in Thm. 8.6 and Cor. 8.7 of [11] in the form that, roughly speaking, algebraic power series are the same as yields of recognizable tree series. The authors use proof techniques, e.g. a theory of fixed points, which assume that the weight semiring is continuous (hence complete) and commutative. However, in some cases these strong assumptions are not necessary to achieve the same result. For instance, we do not need the assumption that $K$ is complete to define the weight of a string in a weighted context-free grammar if, for every $w \in X^*$, the set of derivation trees of $w$ with nonzero weight is finite (cf. the definition of the weighted CF grammar in [8]). The same holds for the yield of a weighted tree language $\Phi$: we do not need the condition that $K$ is complete if, for every $w \in X^*$, the set of $\Sigma X$-trees $\xi$ with $\mathrm{yd}(\xi) = w$ and $\Phi(\xi) \neq 0$ is finite.

In this paper, we extend the above mentioned result of [11] to classes of weighted languages where the weight semiring is not commutative and not necessarily complete. Moreover, using the notions in [14], we will also take into consideration the weighted tree languages recognized by deterministic top-down wta and by deterministic bottom-up wta, as well as weighted languages obtained by taking the yield of local weighted tree languages [12]. For this, we adapt the definition of a weighted CF grammar of [8] to our semiring weighted context-free grammar and call this weighted context-free grammar plain. Moreover, we will introduce the concept of a plain wta and of a plain weighted local system, both as the counterpart of a plain weighted context-free grammar. Then, as the main result of the paper, we will show in Theorem 1 that the following five classes of weighted languages are the same: (i) the class of weighted languages generated by plain weighted context-free grammars, (ii) the class of weighted languages recognized by plain wta, (iii) the class of weighted languages recognized by deterministic and plain top-down wta, (iv) the class of weighted languages recognized by deterministic and plain bottom-up wta, and (v) the class of weighted languages determined by plain weighted local systems.

[1] The weight structure in [8] is a valuation monoid, which is a generalization of a semiring.

# 2 Preliminaries

## 2.1 General concepts

First, let $\mathbb{N}$ be the set of positive integers and $\mathbb{N}_0$ be the set of nonnegative integers. For every $k \in \mathbb{N}$, we define $[k] = \{1, \ldots, k\}$.

An *alphabet* is a finite set $X$ of symbols. We denote by $X^*$ the set of all *words* (or *strings*) over $X$ and by $\varepsilon$ the empty string. The length of a string $w \in X^*$ is denoted by $|w|$. A *language* $L$ (over $X$) is an arbitrary subset of $X^*$.

A *ranked alphabet* is a tuple $(\Sigma, rk)$ where $\Sigma$ is an alphabet and $\mathrm{rk} : \Sigma \to \mathbb{N}_0$ is the rank mapping. For every $k \geq 0$, we define $\Sigma_k = \{\sigma \in \Sigma \mid \mathrm{rk}(\sigma) = k\}$. Sometimes we write $\sigma^{(k)}$ to mean that $\sigma \in \Sigma_k$. Moreover, let $X$ be a set disjoint with $\Sigma$. The set of *terms (or: trees) over* $X$, denoted by $T_\Sigma(X)$, is the smallest set $T$ such that (i) $\Sigma_0 \cup X \subseteq T$ and (ii) if $k \geq 1$, $\sigma \in \Sigma_k$, and $\xi_1, \ldots, \xi_k \in T$, then $\sigma(\xi_1, \ldots, \xi_k) \in T$. We shall abbreviate $T_\Sigma(\emptyset)$ by $T_\Sigma$.

We define the mapping $\mathrm{pos} : T_\Sigma(X) \to \mathcal{P}(\mathbb{N}^*)$ by recursion as follows: (i) for each $y \in (\Sigma_0 \cup X)$ we let $\mathrm{pos}(y) = \{\varepsilon\}$ and (ii) for every $k \geq 1$, $\sigma \in \Sigma^{(k)}$, and $\xi_1, \ldots, \xi_k \in T_\Sigma(X)$ we let $\mathrm{pos}(\sigma(\xi_1, \ldots, \xi_k)) = \{\varepsilon\} \cup \{ip \mid i \in [k], p \in \mathrm{pos}(\xi_i)\}$. For every $\xi \in T_\Sigma(X)$ we call $\mathrm{pos}(\xi)$ the *set of positions in* $\xi$ and, for every $p \in \mathrm{pos}(\xi)$, we define the *label* $\xi(p) \in \Sigma$ *of* $\xi$ *at position* $p$ and the *subtree* $\xi|_p \in T_\Sigma(X)$ *of* $\xi$ *at position* $p$ in the usual way (cf. e.g. [13]). We shall call $\xi(\varepsilon)$ the *root of* $\xi$ and denote it by $\mathrm{rt}(\xi)$.

A monoid $(K, +, 0)$ is *commutative* if $a + b = b + a$ and *zero-sum free* if $a + b = 0$ implies $a = b = 0$ for every $a, b \in K$. We extend the binary summation $+$ to a sum operation $\sum_I : K^I \to K$ for each finite index set $I$ in the usual way. For each finite family $(a_i \mid i \in I)$ of elements of $K$ we write the sum $\sum_I (a_i \mid i \in I)$ also in the form $\sum (a_i \mid i \in I)$ or $\sum_{i \in I} a_i$. Moreover, the monoid $(K, +, 0)$ is *complete* if it has a sum operation $\sum_I : K^I \to K$ for each countable index set $I$ such that this sum coincides with the extension of $+$ when $I$ is finite (for the axioms, see [9, p. 124]). For countable index sets $I$ and families $(a_i \mid i \in I)$ we will also use the notation $\sum (a_i \mid i \in I)$ and $\sum_{i \in I} a_i$ in the same sense as that for finite index sets and families.

A *semiring* is an algebra $(K, +, \cdot, 0, 1)$ which consists of a commutative monoid $(K, +, 0)$, called the additive monoid, and a monoid $(K, \cdot, 1)$, called the multiplicative monoid of the semiring, such that multiplication distributes (from both left and right) over addition, and moreover, $0 \neq 1$ and 0 is absorbing with respect to $\cdot$ (also both from left and right). We call the semiring *zero-sum free* if its additive monoid is zero-sum free and *commutative* if its multiplicative monoid is commutative. Furthermore, the semiring is *complete* if its additive monoid is complete and the generalized distributivity law holds for infinite sums (see [9, p. 124]). An introduction to and some details about semirings can be found e.g. in [17, 18]. As usual, we often denote a semiring by its carrier set.

> *In the rest of this paper* $\Sigma$ *will denote an arbitrary ranked alphabet,* $X$ *will denote an arbitrary alphabet which is disjoint with* $\Sigma$*, and* $K$ *will denote an arbitrary semiring, unless specified otherwise.*

A *$K$-weighted tree language* is a mapping $\Phi : T_\Sigma(X) \to K$. For every $\xi \in T_\Sigma(X)$, the element $\Phi(\xi)$ of $K$ is called the *weight* of $\xi$ (in $\Phi$). Analogously, a *$K$-weighted language* is a mapping $\lambda : X^* \to K$ and, for every $w \in X^*$, the element $\lambda(w)$ of $K$ is called the *weight* of $w$ (in $\lambda$). Sometimes we drop $K$ from $K$-weighted and thus we speak about a weighted (tree) language.

Next, we define the yield of weighted tree languages which satisfies a certain condition. For this, first we define the *yield of a tree* in $T_\Sigma(X)$ by the function $\mathrm{yd}_\Sigma : T_\Sigma(X) \to X^*$ as follows: (i) for every $y \in (\Sigma_0 \cup X)$ let $\mathrm{yd}_\Sigma(y) = \varepsilon$ if $y \in \Sigma_0$ and $\mathrm{yd}_\Sigma(y) = y$ if $y \in X$, and (ii) for every $\xi = \sigma(\xi_1, \ldots, \xi_k)$, where $k \geq 1$, we define $\mathrm{yd}_\Sigma(\xi) = \mathrm{yd}_\Sigma(\xi_1) \ldots \mathrm{yd}_\Sigma(\xi_k)$. Hence, we have $\mathrm{yd}_\Sigma^{-1}(w) = \{\xi \in T_\Sigma(X) \mid \mathrm{yd}_\Sigma(\xi) = w\}$ for every $w \in X^*$.

Now let $\Phi : T_\Sigma(X) \to K$ be a weighted tree language. We call $\Phi$ *summable for yield* (or: *summable*) if the semiring $K$ is complete or the set

$$T_\Phi(w) = \{\xi \in \mathrm{yd}_\Sigma^{-1}(w) \mid \Phi(\xi) \neq 0\}$$

is finite for every $w \in X^*$. If $\Phi$ is summable, then we define the *yield of* $\Phi$ to be the weighted language $\mathrm{yd}(\Phi) : X^* \to K$ by

$$\mathrm{yd}(\Phi)(w) = \sum_{\xi \in T_\Phi(w)} \Phi(\xi)$$

for every $w \in X^*$, where $\sum$ denotes the extension of the addition of $K$. (The fact that $\Phi$ is summable guarantees that the above sum is well-defined.) Moreover, for a class $C(K)$ of summable $K$-weighted languages we define $\mathrm{yd}'(C(K)) = \{\mathrm{yd}(\Phi) \mid \Phi \in C(K)\}$ and we will write yd for $\mathrm{yd}'$ in the rest of the paper.

## 2.2 Weighted context-free languages

Weighted context-free grammars over semirings were introduced in [5] (see also [24, 19]). Recently, a Chomsky-Schützenberger theorem was proved for weighted context-free grammars over tree valuation monoids in [8]. We follow the idea of [8] to define the semantics of a weighted context-free grammar, but we will use semirings as weight structures.

A *$K$-weighted context-free grammar* (or $\mathrm{CF}(K)$-grammar for short) is a tuple $\mathcal{G} = (N, X, Z, P, \mathrm{wt})$, where $N$ and $X$ are alphabets (*nonterminals* and *terminals*, respectively) such that $N \cap X = \emptyset$, $Z \in N$ (*initial nonterminal*), $P$ is a finite set of *rules* of the form $A \to \alpha$, where $A \in N$ and $\alpha \in (N \cup X)^*$, and $\mathrm{wt}\colon P \to K$ is a mapping (*weight assignment*). Given a rule $r = (A \to \alpha)$, we call the nonterminal $A$ the *left-hand side of* $r$ and denote it by $\mathrm{lhs}(r)$.

The semantics of a weighted context-free grammar is defined in [8] in terms of leftmost derivations. Here, we follow an equivalent approach and use derivation trees in the sense of [16, Sect. 3.1]. In fact, we will treat $P$ as a ranked alphabet by letting $\mathrm{rk}(r) = |\alpha|$ for every $r = (A \to \alpha) \in P$ and we will denote this ranked alphabet by $\bar{P}$. Hence $\bar{P}_k = \{(A \to \alpha) \in P \mid |\alpha| = k\}$ for every $k \geq 0$.

We can extend the mapping wt to trees in $T_{\bar{P}}(X)$ by defining the mapping $\mathrm{wt}' : T_{\bar{P}}(X) \to K$ as follows. For every $\zeta \in T_{\bar{P}}(X)$,

(i) if $\zeta = r$ for some rule $r \in \bar{P}_0$, then $\mathrm{wt}'(\zeta) = \mathrm{wt}(r)$,

(ii) if $\zeta \in X$, then $\mathrm{wt}'(\zeta) = 1$, and

(iii) if $\zeta = r(\zeta_1, \ldots, \zeta_k)$, for some $k \geq 1$, $r \in \bar{P}_k$, and $\zeta_1, \ldots, \zeta_k \in T_{\bar{P}}(X)$, then $\mathrm{wt}'(\zeta) = \mathrm{wt}'(\zeta_1) \cdot \ldots \cdot \mathrm{wt}'(\zeta_k) \cdot \mathrm{wt}(r)$ (where $\cdot$ is the multiplication of $K$).

We note that $\mathrm{wt}'$ is a $K$-weighted tree language, thus we may call $\mathrm{wt}'(\zeta)$ the weight of $\zeta$. From now on, we write wt for $\mathrm{wt}'$.

Next, we define derivation trees as certain trees in $T_{\bar{P}}(X)$. Formally, for every $w \in X^*$, we define the set $D_{\mathcal{G}}(w)$ of *derivation trees of* $w$ such that, for every $\zeta \in T_{\bar{P}}(X)$, we have $\zeta \in D_{\mathcal{G}}(w)$, if and only if

- $\mathrm{lhs}(\mathrm{rt}(\zeta)) = Z$ and $\mathrm{yd}_{\bar{P}}(\zeta) = w$,
- for every $p \in \mathrm{pos}(\zeta)$ with $\zeta(p) = (A \to \alpha_1 \ldots \alpha_k)$ for some $k \geq 1$ and $\alpha_1, \ldots, \alpha_k \in (N \cup X)$, we have $\zeta(pi) = y_i$, where

$$y_i = \begin{cases} \alpha_i & \text{if } \alpha_i \in X \\ \text{a rule } r_i \in \bar{P} \text{ with } \mathrm{lhs}(r_i) = \alpha_i & \text{if } \alpha_i \in N, \end{cases}$$

  for every $1 \leq i \leq k$.

The following concept was suggested by [8]. However, we will use a new name to identify the defined class of weighted context-free grammars. We call $\mathcal{G}$ *plain* if the semiring $K$ is complete or the set $\{\zeta \in D_{\mathcal{G}}(w) \mid \mathrm{wt}(\zeta) \neq 0\}$ is finite for every $w \in X^*$. In this case we define the *weighted language generated by* $\mathcal{G}$ to be the $K$-weighted language $\lambda_{\mathcal{G}} : X^* \to K$ given for every $w \in X^*$ by

$$\lambda_{\mathcal{G}}(w) = \sum_{\zeta \in D_{\mathcal{G}}(w), \mathrm{wt}(\zeta) \neq 0} \mathrm{wt}(\zeta).$$

The class of weighted languages generated by plain $\mathrm{CF}(K)$-grammars is denoted by $\mathrm{CFL}_{\mathrm{p}}(K)$.

**Example 1.** It is known that the language $L = \{w \in \{0,1\}^* \mid |w|_0 = |w|_1\}$ is context-free. It can be generated, for instance, by the context-free grammar

$$r_1 : S \to SS,\ r_2 : S \to 0S1,\ r_3 : S \to 1S0,\ \text{and } r_4 : S \to \varepsilon\ .$$

This grammar is ambiguous; that is, there are words in $L$ which have more than one derivation tree.

Now we will consider the tropical semiring $\mathsf{Trop} = (\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$. It is well known that $\mathsf{Trop}$ is complete. Then we define the $\mathrm{CF}(\mathsf{Trop})$-grammar $\mathcal{G} = (\{S\}, X, S, P, \mathrm{wt})$, where $X = \{0,1\}$, $P = \{r_1, r_2, r_3, r_4\}$, $\mathrm{wt}(r_1) = \mathrm{wt}(r_2) = \mathrm{wt}(r_3) = 0$, and $\mathrm{wt}(r_4) = 1$. The grammar $\mathcal{G}$ is plain, because $\mathsf{Trop}$ is complete. In Figure 1, we show two trees in $T_{\bar{P}}(X)$, where the rank of $r_1, r_2, r_3$, and $r_4$ in $\bar{P}$ is $2, 3, 3$, and $0$, respectively. The first tree (from left to right) is not a derivation

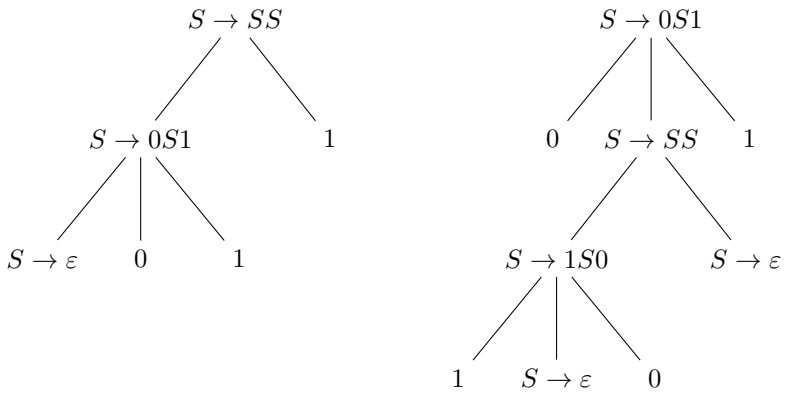



Figure 1: Two trees in $T_{\bar{P}}(X)$ of Example 1.

tree of any $w \in X^*$, while the second one is a derivation tree of 0101, i.e. it is in $D_{\mathcal{G}}(0101)$. The weight of the first tree is 1 and the weight of the second one is 2.

Now let $w \in \Sigma^*$. It is clear that for every $\zeta \in T_{\bar{P}}(X)$, the weight of $\zeta$ is the number of the occurrences of $r_4$ (roughly speaking, the number of erasing rules) in $\zeta$. Let us denote this number by $\#_{\mathrm{ers}}(\zeta)$. Moreover,

$$\lambda_{\mathcal{G}}(w) = \min(\mathrm{wt}(\zeta) \mid \zeta \in D_{\mathcal{G}}(w), \mathrm{wt}(\zeta) \neq \infty) = \min(\#_{\mathrm{ers}}(\zeta) \mid \zeta \in D_{\mathcal{G}}(w)).$$

## 2.3 Recognizable weighted tree languages

A *$K$-weighted tree automaton with yield alphabet* (or $K$-wta for short) is a tuple $\mathcal{A} = (Q, \Sigma, X, \delta, \kappa)$ where $Q$ is a finite nonempty set, the *set of states*, $\Sigma$ is the *ranked input alphabet*, $X$ is the *yield alphabet*, $\delta = (\delta_k \mid k \in \mathbb{N}_0)$ is a *family of transition mappings*[2] such that

$$\delta_k : Q^k \times \Sigma_k \times Q \to K \text{ for } k \geq 1 \text{ and } \delta_0 : (\Sigma_0 \cup X) \times Q \to K,$$

and $\kappa : Q \to K$ is the *root weight mapping*.

For every $k \in \mathbb{N}$ we call an element $(q_1 \dots q_k, \sigma, q) \in Q^k \times \Sigma_k \times Q$ a *transition*, and call $\delta_k(q_1 \dots q_k, \sigma, q) \in K$ the *weight of that transition*. (Here and in the rest of the paper, we abbreviate $(q_1, \dots, q_k)$ by $q_1 \dots q_k$.)

Let $\xi \in T_\Sigma(X)$. A *run of $\mathcal{A}$ on $\xi$* is a mapping $\omega : \mathrm{pos}(\xi) \to Q$. The set of all runs of $\mathcal{A}$ on $\xi$ is denoted by $R_{\mathcal{A}}(\xi)$. For every $\omega \in R_{\mathcal{A}}(\xi)$ and $p \in \mathrm{pos}(\xi)$, the run $\omega|_p$ of $\mathcal{A}$ on $\xi|_p$ is defined by $\omega|_p(p') = \omega(pp')$ for every $p' \in \mathrm{pos}(\xi|_p)$. Now we define the *weight of a run* $\omega \in R_{\mathcal{A}}(\xi)$ to be an element $\delta^*(\omega)$ of $K$ by induction as follows:

[2]In the literature $\delta$ is also called a *tree representation* and $\delta_k$ is given as a mapping of type $\Sigma_k \to S^{Q^k \times Q}$.

- if $\xi = y \in (\Sigma_0 \cup X)$, then $\delta^*(\omega) = \delta_0(y, \omega(\varepsilon))$,
- if $\xi = \sigma(\xi_1, \ldots, \xi_k)$ for some $k \geq 1$, then

$$\delta^*(\omega) = \delta^*(\omega|_1) \cdot \ldots \cdot \delta^*(\omega|_k) \cdot \delta_k(\omega(1) \ldots \omega(k), \sigma, \omega(\varepsilon)),$$

where $\cdot$ is the product of the semiring $K$. (Note that $\omega|_i \in R_{\mathcal{A}}(\xi_i)$ because $\xi_i = \xi|_i$ for every $1 \leq i \leq k$).

The $K$-weighted tree language $\|\mathcal{A}\| : T_\Sigma(X) \to K$ *recognized* by $\mathcal{A}$ is defined by

$$\|\mathcal{A}\|(\xi) = \sum_{\omega \in R_{\mathcal{A}}(\xi)} \delta^*(\omega) \cdot \kappa(\omega(\varepsilon))$$

for every $\xi \in T_\Sigma(X)$. An introduction to the theory of wta over semirings and some results can be found in [4], [11], and [13].

**Example 2.** (Cf. [3, Example 3.3]) We consider the arctic semiring $\mathsf{Arct} = (\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)$ and construct the wta $\mathcal{A} = (Q, \Sigma, X, \delta, \kappa)$ which recognizes the weighted tree language $\text{height} : T_\Sigma(X) \to \mathbb{N}$, where $\text{height}(\xi) = \max\{|w| \mid w \in \text{pos}(\xi)\}$. For this, let $Q = \{p_1, p_2\}$, $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$, $X = \{x_1, x_2\}$. Furthermore, let

$$\begin{array}{lllll} \delta_0(y, p_1) & = & \delta_0(y, p_2) & = & 0, \text{ for all } y \in (\Sigma_0 \cup X), \\ \delta_2(p_1 p_2, \sigma, p_1) & = & \delta_2(p_2 p_1, \sigma, p_1) & = & 1, \\ \delta_2(p_2 p_2, \sigma, p_2) & = & 0, & & \end{array}$$

and for every other transition $(q_1 q_2, \sigma, q)$ we have $\delta_2(q_1 q_2, \sigma, q) = -\infty$. Lastly, let $\kappa(p_1) = 0$ and $\kappa(p_2) = -\infty$.

Intuitively, $\mathcal{A}$ works as follows. For every input tree $\xi$ and run $\omega \in R_{\mathcal{A}}(\xi)$,

- if $\omega$ assigns $p_1$ to each position in a path from the root to a leaf of $\xi$ (in particular, to the root and to that leaf of $\xi$) and assigns $p_2$ to every other position in $\xi$, then the weight of $\omega$ is equal to the length of that path,
- if $\omega$ assigns $p_2$ to each position in $\xi$, then the weight of $\omega$ is 0, and
- in every other case, the weight of $\omega$ is $-\infty$.

Hence,

$$\begin{aligned} &\max\big(\delta^*(\omega) \mid \omega \in R_{\mathcal{A}}(\xi), \omega(\varepsilon) = p_1\big) = \text{height}(\xi) \text{ and} \\ &\max\big(\delta^*(\omega) \mid \omega \in R_{\mathcal{A}}(\xi), \omega(\varepsilon) = p_2\big) = 0, \end{aligned}$$

for every $\xi \in T_\Sigma(X)$. Thus,

$$\begin{aligned} &\|\mathcal{A}\|(\xi) = \max(\delta^*(\omega) + \kappa(\omega(\varepsilon)) \mid \omega \in R_{\mathcal{A}}(\xi)) = \\ &\max\big(\max(\delta^*(\omega) + \kappa(p_1) \mid \omega \in R_{\mathcal{A}}(\xi), \omega(\varepsilon) = p_1), \\ &\qquad \max(\delta^*(\omega) + \kappa(p_2) \mid \omega \in R_{\mathcal{A}}(\xi), \omega(\varepsilon) = p_2)\big) = \\ &\max\big(\text{height}(\xi) + 0, 0 + (-\infty)\big) = \text{height}(\xi). \end{aligned}$$

A $K$-wta $\mathcal{A} = (Q, \Sigma, X, \delta, \kappa)$ is *bottom-up deterministic* (or bu-deterministic) if for every $y \in (\Sigma_0 \cup X)$, there is at most one $q \in Q$ such that $\delta_0(y, q) \neq 0$, and for every $k \geq 1$, $\sigma \in \Sigma_k$, and $w \in Q^k$ there is at most one $q \in Q$ such that $\delta_k(w, \sigma, q) \neq 0$. If this is the case, then for every input tree $\xi \in T_\Sigma(X)$, there is at most one $\omega \in R_\mathcal{A}(\xi)$ such that $\delta^*(\omega) \neq 0$. Thus $\|\mathcal{A}\|(\xi) = \delta^*(\omega) \cdot \kappa(\omega(\varepsilon))$, if $\omega$ is the only element of $R_\mathcal{A}(\xi)$ with $\delta^*(\omega) \neq 0$ and $\|\mathcal{A}\|(\xi) = 0$ if there is no such element in $R_\mathcal{A}(\xi)$.

Moreover, $\mathcal{A}$ is *top-down deterministic* (or td-deterministic) if the set $\{q \in Q \mid \kappa(q) \neq 0\}$ is a singleton, for every $y \in (\Sigma_0 \cup X)$, there is at most one $q \in Q$ such that $\delta_0(y, q) \neq 0$, and for every $k \geq 1$, $\sigma \in \Sigma_k$, and $q \in Q$ there is at most one $w \in Q^k$ such that $\delta_k(w, \sigma, q) \neq 0$. In this case, for every $q \in Q$ and $\xi \in T_\Sigma(X)$, there is at most one $\omega \in R_\mathcal{A}(\xi)$ with $\omega(\varepsilon) = q$ and $\delta^*(\omega) \neq 0$. Hence the formula for $\|\mathcal{A}\|(\xi)$ can be simplified in the same way as for a bu-deterministic $K$-wta. Let us mention that for both kinds of deterministic $K$-wta, the addition $+$ of $K$ is not used to the compute $\|\mathcal{A}\|$.

A $K$-weighted tree language $\Phi : T_\Sigma(X) \to K$ is *recognizable* (bu-*deterministically recognizable*, td-*deterministically recognizable*) if there is a $K$-wta (resp. bu-deterministic $K$-wta, td-deterministic $K$-wta) $\mathcal{A}$ such that $\Phi = \|\mathcal{A}\|$. The class of all summable and recognizable $K$-weighted tree languages is denoted by $\mathrm{Rec_s}(K)$. The notations bud-$\mathrm{Rec_s}(K)$ and tdd-$\mathrm{Rec_s}(K)$ are introduced in an analogous way.

## 2.4 Local weighted tree languages

Local weighted tree languages were introduced in [12]. Here, we give a slightly more general definition by using a yield alphabet $X$ in order to be able to handle yields of local weighted tree languages.

We introduce the family $\mathrm{Fork}(\Sigma, X) = (\mathrm{Fork}_k(\Sigma, X) \mid k \geq 0)$ of sets, where

$$\mathrm{Fork}_k(\Sigma, X) = (\Sigma \cup X)^k \times \Sigma_k \text{ for } k \geq 1 \text{ and } \mathrm{Fork}_0(\Sigma, X) = \Sigma_0 \cup X.$$

We write the elements of $\mathrm{Fork}_k(\Sigma, X)$, $k \geq 1$ in the form $(y_1 \dots y_k, \sigma)$ and call them $(\Sigma, X)$-*forks*. A fork $(y_1 \dots y_k, \sigma)$ occurs in a tree if the tree has a $\sigma$-node of which the $k$ children are labeled by $y_1, \dots, y_k$ from the left to right.

A $K$-*weighted local system* (or $K$-wls for short) is a system $\mathcal{L} = (\Sigma, X, \varphi, \rho)$, where $\varphi$ is a family of mappings $(\varphi_k \mid k \geq 0)$ with

$$\varphi_k : \mathrm{Fork}_k(\Sigma \cup X) \to K, \text{ and } \rho : (\Sigma \cup X) \to K$$

is another mapping. Intuitively, we associate a weight, i.e., an element of $K$ with each fork and also with each symbol in $\Sigma \cup X$. Note that this weight may be 0.

Next, we define the $K$-weighted tree language determined by $\mathcal{L}$. For this, we extend $\varphi$ to the mapping $\varphi' : T_\Sigma(X) \to K$ defined by induction as follows:

(i) $\varphi'(y) = \varphi_0(y)$ for every $y \in (\Sigma_0 \cup X)$,

(ii) $\varphi'(\sigma(\xi_1, \dots, \xi_k)) = \varphi'(\xi_1) \cdot \dots \cdot \varphi'(\xi_k) \cdot \varphi_k(\mathrm{rt}(\xi_1) \dots \mathrm{rt}(\xi_k), \sigma)$ for every $k \geq 1$, $\sigma \in \Sigma_k$, and $\xi_1, \dots, \xi_k \in T_\Sigma(X)$.

In the following we write $\varphi$ for $\varphi'$. The *$K$-weighted tree language* $\|\mathcal{L}\| : T_\Sigma(X) \to K$ *determined by* $\mathcal{L}$ is defined by $\|\mathcal{L}\|(\xi) = \varphi(\xi) \cdot \rho(\mathrm{rt}(\xi))$ for every $\xi \in T_\Sigma(X)$. As for deterministic $K$-wta, the operation $+$ of $K$ is not used in the definition of $\|\mathcal{L}\|$.

Thus, $\varphi(\xi)$ is the (semiring) product of the weights associated with the forks in $\xi$. The order of the factors is the postorder of the nodes of $\xi$. Also, the weight $\|\mathcal{L}\|(\xi)$ of $\xi$ is the product of $\varphi(\xi)$ and the weight associated to the root of $\xi$.

**Example 3.** We consider again the ranked alphabet $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$, the set $X = \{x_1, x_2\}$ and the semiring $\mathsf{Arct}$. We define the $\mathsf{Arct}$-wls $\mathcal{L} = (\Sigma, X, \varphi, \rho)$ by

- $\varphi_2(y\alpha, \sigma) = 1$, $\varphi_2(yz, \sigma) = 0$ for all $y, z \in (\Sigma \cup X)$ with $z \neq \alpha$, and $\varphi_0(y) = 0$ for all $y \in (\Sigma_0 \cup X)$, and
- $\rho(y) = 0$ for all $y \in (\Sigma \cup X)$.

It should be clear that $\|\mathcal{L}\|(\xi)$ is the number of the occurrences of the pattern $\sigma(\_, \alpha)$ in $\xi$ for every $\xi \in T_\Sigma(X)$, where '$\_$' is a placeholder which may be filled by any element of $\Sigma \cup X$. We note that in [13, Example 3.4] a wta is given over the semiring of natural numbers which recognizes $\|\mathcal{L}\|$ (with the difference being that there $X = \emptyset$).

A $K$-weighted tree language $\Phi : T_\Sigma(X) \to K$ is called *local* if there is a $K$-wls $\mathcal{L}$ such that $\Phi = \|\mathcal{L}\|$.

# 3 The results

Now, we will introduce plain wta and plain wls and define weighted languages recognizable by plain wta and determined by plain wls, respectively. We relate the class of weighted languages generated by plain weighted context-free grammars, the class of weighted languages recognizable by plain wta, and the class of weighted languages determined by plain wls.

We say that a $K$-wta $\mathcal{A} = (Q, \Sigma, X, \delta, \kappa)$ is *plain* if $K$ is complete or, for every $w \in \Sigma^*$, the set

$$U_\mathcal{A}(w) = \{\xi \in \mathrm{yd}_\Sigma^{-1}(w) \mid \exists(\omega \in R_\mathcal{A}(\xi)) : \delta^*(\omega) \cdot \kappa(\omega(\varepsilon)) \neq 0\}$$

is finite.

**Lemma 1.** Let $\mathcal{A} = (Q, \Sigma, X, \delta, \kappa)$ be a $K$-wta.

(1) If $\mathcal{A}$ is plain, then $\|\mathcal{A}\|$ is summable and

$$\mathrm{yd}(\|\mathcal{A}\|)(w) = \sum_{\xi \in U_\mathcal{A}(w), \omega \in R_\mathcal{A}(\xi)} \delta^*(\omega) \cdot \kappa(\omega(\varepsilon))$$

for every $w \in \Sigma^*$.

(2) If $K$ is zero-sum free and $\|\mathcal{A}\|$ is summable, then $\mathcal{A}$ is plain.

(3) If $\mathcal{A}$ is bu-deterministic and $\|\mathcal{A}\|$ is summable, then $\mathcal{A}$ is plain. The same holds when we replace bu-deterministic by td-deterministic.

*Proof.* Let $w \in \Sigma^*$. It is obvious that

$$T_{\|\mathcal{A}\|}(w) =$$

$$\{\xi \in \mathrm{yd}_\Sigma^{-1}(w) \mid \|\mathcal{A}\|(\xi) \neq 0\} = \{\xi \in \mathrm{yd}_\Sigma^{-1}(w) \mid \left(\sum_{\omega \in R_\mathcal{A}(\xi)} \delta^*(\omega) \cdot \kappa(\omega(\varepsilon))\right) \neq 0\} \subseteq$$

$$\{\xi \in \mathrm{yd}_\Sigma^{-1}(w) \mid \exists(\omega \in R_\mathcal{A}(\xi)) : \delta^*(\omega) \cdot \kappa(\omega(\varepsilon)) \neq 0\} = U_\mathcal{A}(w).$$

Now, we will prove (1). Since $\mathcal{A}$ is plain, the set $U_\mathcal{A}(w)$ is finite. Thus $T_{\|\mathcal{A}\|}(w)$ is also finite, hence $\|\mathcal{A}\|$ is summable. Moreover,

$$\mathrm{yd}(\|\mathcal{A}\|)(w) = \sum_{\xi \in T_{\|\mathcal{A}\|}(w)} \|\mathcal{A}\|(\xi) = \sum_{\xi \in T_{\|\mathcal{A}\|}(w)} \left(\sum_{\omega \in R_\mathcal{A}(\xi)} \delta^*(\omega) \cdot \kappa(\omega(\varepsilon))\right) =$$

$$\sum_{\xi \in U_\mathcal{A}(w)} \left(\sum_{\omega \in R_\mathcal{A}(\xi)} \delta^*(\omega) \cdot \kappa(\omega(\varepsilon))\right) = \sum_{\xi \in U_\mathcal{A}(w), \omega \in R_\mathcal{A}(\xi)} \delta^*(\omega) \cdot \kappa(\omega(\varepsilon)),$$

where the third equality holds because for every $\xi \in (U_\mathcal{A}(w) \setminus T_{\|\mathcal{A}\|}(w))$ the corresponding sum is 0 and the fourth one holds because summation is associative and commutative in $K$.

To prove (2), we assume that $K$ is zero-sum free and that $\|\mathcal{A}\|$ is summable. Due to the fact that $K$ is zero-sum free, $\subseteq$ becomes an equality and therefore $T_{\|\mathcal{A}\|}(w) = U_\mathcal{A}(w)$. Since $T_{\|\mathcal{A}\|}(w)$ is finite, the set $U_\mathcal{A}(w)$ is also finite and hence $\mathcal{A}$ is plain.

Statement (3) follows from the fact that, by the remarks we made on the runs of bu-deterministic wta and td-deterministic wta, $\subseteq$ becomes an equality and so we have again that $T_{\|\mathcal{A}\|}(w) = U_\mathcal{A}(w)$. □

Let $\mathrm{Rec_p}(K)$ be the class of all $K$-weighted tree languages which are recognizable by a plain $K$-wta. The notations bud-$\mathrm{Rec_p}(K)$ and tdd-$\mathrm{Rec_p}(K)$ are introduced in an analogous way.

Let $\mathcal{A}$ be a plain $K$-wta. Then we call $\mathrm{yd}(\|\mathcal{A}\|)$ *the weighted language recognized by* $\mathcal{A}$ and denote it by $\lambda_\mathcal{A}$. Note that

$$\lambda_\mathcal{A} = \sum_{\xi \in U_\mathcal{A}(w), \omega \in R_\mathcal{A}(\xi)} \delta^*(\omega) \cdot \kappa(\omega(\varepsilon)).$$

The next statements immediately follow from Lemma 1.

**Corollary 1.** (1) $\mathrm{yd}(\mathrm{Rec_p}(K)) \subseteq \mathrm{yd}(\mathrm{Rec_s}(K))$.

(2) If $K$ is zero-sum free, then $\mathrm{yd}(\mathrm{Rec_p}(K)) = \mathrm{yd}(\mathrm{Rec_s}(K))$.

(3) $\mathrm{yd}(\text{bud-}\mathrm{Rec_p}(K)) = \mathrm{yd}(\text{bud-}\mathrm{Rec_s}(K))$.

(4) $\mathrm{yd}(\text{tdd-}\mathrm{Rec_p}(K)) = \mathrm{yd}(\text{tdd-}\mathrm{Rec_s}(K))$.

It is an open question whether there is a semiring $K$ such that $\mathrm{yd}(\mathrm{Rec_s}(K)) \setminus \mathrm{yd}(\mathrm{Rec_p}(K)) \neq \emptyset$. However, we can prove the following weaker statement.

**Lemma 2.** There is a semiring $K$ and a $K$-wta $\mathcal{A}$ which is not plain such that $\|\mathcal{A}\|$ is summable.

*Proof.* We consider the semiring $(\mathbb{Z}, +, \cdot, 0, 1)$ of integers. We note that $\mathbb{Z}$ is not zero-sum free. Moreover, we define the $\mathbb{Z}$-wta $\mathcal{A} = (Q, \Sigma, X, \delta, \kappa)$, where $Q = \{p, q, r\}$, $\Sigma = \Sigma_1 = \{\gamma\}$, $X = \{x\}$. Moreover,

- $\delta_0(x, p) = -1$, $\delta_0(x, q) = \delta_0(x, r) = 1$,
- $\delta_1(p, \gamma, p) = \delta_1(q, \gamma, q) = 1$ and $\delta_1(s, \gamma, t) = 0$ for every other combination $s, t \in Q$,
- $\kappa(p) = \kappa(q) = \kappa(r) = 1$.

There are three runs $\omega_p$, $\omega_q$, and $\omega_r$ on the input tree $x$, which are defined by $\omega_p(\varepsilon) = p$, $\omega_q(\varepsilon) = q$, and $\omega_r(\varepsilon) = r$. For these runs, we have

$$\delta^*(\omega_p) \cdot \kappa(p) + \delta^*(\omega_q) \cdot \kappa(q) + \delta^*(\omega_r) \cdot \kappa(r) = (-1) \cdot 1 + 1 \cdot 1 + 1 \cdot 1 = 1,$$

hence $\|\mathcal{A}\|(x) = 1$. For each $n \geq 1$, there are two runs $\omega_{p,n}$ and $\omega_{q,n}$ with nonzero weight on the tree $\gamma^n(x)$. The run $\omega_{p,n}$ associates $p$ with each position in $\gamma^n(x)$, and the run $\omega_{q,n}$ is defined analogously. For these runs, we have

$$\delta^*(\omega_{p,n}) \cdot \kappa(p) + \delta^*(\omega_{q,n}) \cdot \kappa(q) = (-1) \cdot 1 + 1 \cdot 1 = 0,$$

hence $\|\mathcal{A}\|(\gamma^n(x)) = 0$. This means that $T_{\|\mathcal{A}\|}(x) = \{x\}$ and $T_{\|\mathcal{A}\|}(w) = \emptyset$ for every $w \in X^*$ with $w \neq x$. Hence $\mathcal{A}$ is summable.

However, for every $\xi \in T_\Sigma(X)$, we have $\mathrm{yd}_\Sigma(\xi) = x$ and there is a run $\omega$ on $\xi$ such that $\delta^*(\omega) \cdot \kappa(\omega(\varepsilon)) \neq 0$. Hence the set $U_\mathcal{A}(x)$ is infinite and thus $\mathcal{A}$ is not plain. □

Now we turn to local weighted tree languages and the weighted languages determined by them.

We call a $K$-weighted local system $\mathcal{L} = (\Sigma, X, \varphi, \rho)$ *plain* if $K$ is complete or the set $\{\xi \in \mathrm{yd}_\Sigma^{-1}(w) \mid \varphi(\xi) \cdot \rho(\mathrm{rt}(\xi)) \neq 0\}$ is finite for every $w \in X^*$. It follows immediately from the corresponding definitions that a $K$-wls $\mathcal{L}$ is plain if and only if $\|\mathcal{L}\|$ is summable. For a plain $K$-wls $\mathcal{L}$, we call $\mathrm{yd}(\|\mathcal{L}\|)$ *the weighted language determined by* $\mathcal{L}$ and denote it by $\lambda_\mathcal{L}$. We denote by $\mathrm{Loc_p}(K)$ the class of weighted tree languages determined by plain $K$-weighted local systems.

**Proposition 1.** [12, Lm. 1] $\mathrm{Loc_p}(K) \subseteq \text{bud-}\mathrm{Rec_p}(K)$.

*Proof.* The construction used in the proof of [14, Thm. II. 9.4] (see also Lemma 1 of [12]) can be naturally extended to the yield alphabet. Indeed, let $\mathcal{L} = (\Sigma, X, \varphi, \rho)$ be a $K$-wls and construct the $K$-wta $\mathcal{A} = (Q, \Sigma, X, \delta, \kappa)$ in the following way. Let $Q = \{\overline{z} \mid z \in (\Sigma \cup X)\}$ and, for every $y \in (\Sigma_0 \cup X)$ and $z \in (\Sigma \cup X)$, let $\delta_0(y, \overline{z}) = \varphi_0(y)$, if $z = y$ and let $\delta_0(y, \overline{z}) = 0$, otherwise. Furthermore, for every $k \geq 1, z_1 \dots z_k \in (\Sigma \cup X)^k, \sigma \in \Sigma_k$, and $z \in (\Sigma \cup X)$, let

$$\delta_k(\overline{z_1} \dots \overline{z_k}, \sigma, \overline{z}) = \begin{cases} \varphi_k(z_1 \dots z_k, \sigma) & \text{if } z = \sigma \\ 0 & \text{otherwise.} \end{cases}$$

Lastly, for every $\sigma \in \Sigma$, let $\kappa(\overline{\sigma}) = \rho(\sigma)$.

It is easy to see that $\mathcal{A}$ is bu-deterministic. Now let $\xi \in T_\Sigma(X)$ and $\omega_\xi \in R_\mathcal{A}(\xi)$ be the run defined by $\omega_\xi(p) = \overline{\xi(p)}$, for every $p \in \mathrm{pos}(\xi)$. It can be readily seen by induction on $\xi$ that $\delta^*(\omega_\xi) = \varphi(\xi)$. Moreover, for every run $\omega \in R_\mathcal{A}(\xi)$ with $\omega \neq \omega_\xi$, we have $\delta^*(\omega) = 0$. Then, for every $\xi$ in $T_\Sigma(X)$, we get that

$$\|\mathcal{A}\|(\xi) = \sum_{\omega \in \mathcal{R}_\mathcal{A}(\xi)} \delta^*(\omega) \cdot \kappa(\omega(\varepsilon)) = \delta^*(\omega_\xi) \cdot \kappa(\omega_\xi(\varepsilon)) = \\ \delta^*(\omega_\xi) \cdot \kappa(\overline{\xi(\varepsilon)}) = \varphi(\xi) \cdot \rho(\mathrm{rt}(\xi)) = \|\mathcal{L}\|(\xi).$$

Now assume that $\mathcal{L}$ is plain. By our above remark, $\|\mathcal{L}\|$ is summable. Hence $\|\mathcal{A}\|$ is also summable. Since $\mathcal{A}$ is bu-deterministic, by Lemma 1(3) we obtain that $\mathcal{A}$ is plain. □

Now we have all the concepts available to state the main result of this paper.

**Theorem 1.** *For each weighted language $\lambda : T_\Sigma(X) \to K$, the following five statements are equivalent:*

*(1) $\lambda$ can be generated by a plain $\mathrm{CF}(K)$-grammar,*

*(2) $\lambda$ can be determined by a plain $K$-wls,*

*(3) $\lambda$ can be recognized by a plain and bottom-up deterministic $K$-wta,*

*(4) $\lambda$ can be recognized by a plain and top-down deterministic $K$-wta,*

*(5) $\lambda$ can be recognized by a plain $K$-wta.*

*If $K$ is zero-sum free, then the following can be added to the list:*

*(6) $\lambda$ can be recognized by a $K$-wta $\mathcal{A}$ such that $\|\mathcal{A}\|$ is summable.*

*Proof.* The proof of the first statement is that (1) ⇒ (2) by Lemma 3, (2) ⇒ (3) by Proposition 1, (1) ⇒ (4) by Lemma 4, (3), (4) ⇒ (5) by definition, and finally (5) ⇒ (1) by Lemma 6. The second statement follows from Corollary 1(2). □

**Lemma 3.** $\mathrm{CFL}_\mathrm{p}(K) \subseteq \mathrm{yd}(\mathrm{Loc}_\mathrm{p}(K))$.

*Proof.* Let $\mathcal{G} = (N, X, Z, P, \mathrm{wt})$ be a plain CF($K$)-grammar. We define the $K$-wls $\mathcal{L} = (\bar{P}, X, \varphi, \rho)$, where

- $\bar{P}$ is the ranked alphabet defined in Section 2.2,
- for every $k \geq 1$, the mapping $\varphi_k : \mathrm{Fork}_k(\bar{P} \cup X) \to K$ is defined by $\varphi_k(y_1 \dots y_k, r) = \mathrm{wt}(r)$ if $r = (A \to \alpha_1 \dots \alpha_k)$ for some $k \geq 1$ and $\alpha_1, \dots, \alpha_n \in (N \cup X)$, and

$$y_i = \begin{cases} \alpha_i & \text{if } \alpha_i \in X \\ \text{a rule } r_i \in P \text{ with } \mathrm{lhs}(r_i) = \alpha_i & \text{if } \alpha_i \in N, \end{cases}$$

  for every $1 \leq i \leq k$; and $\varphi_k(y_1 \dots y_k, r) = 0$ in every other case,
- the mapping $\varphi_0 : \mathrm{Fork}_0(\bar{P} \cup X) \to K$ is defined by $\varphi_0(r) = \mathrm{wt}(r)$ for every $r \in \bar{P}_0$ and $\varphi_0(x) = 1$ for every $x \in X$,
- the root mapping $\rho : (\bar{P} \cup X) \to K$ is defined, for every $y \in (\bar{P} \cup X)$ by $\rho(y) = 1$ if $y \in P$ with $\mathrm{lhs}(y) = Z$ and $\rho(y) = 0$ in every other case.

Let $w \in X^*$. Due to the construction, $D_{\mathcal{G}}(w) \subseteq \mathrm{yd}_{\bar{P}}^{-1}(w)$ and

$$\varphi(\zeta) \cdot \rho(\mathrm{rt}(\zeta)) = \begin{cases} \mathrm{wt}(\zeta) & \text{if } \zeta \in D_{\mathcal{G}}(w) \\ 0 & \text{otherwise} \end{cases}$$

for every $\zeta \in \mathrm{yd}_{\bar{P}}^{-1}(w)$. If $K$ is not complete, then the set $\{\zeta \in D_{\mathcal{G}}(w) \mid \mathrm{wt}(\zeta) \neq 0\}$ is finite because $\mathcal{G}$ is plain. Thus the set $\{\zeta \in \mathrm{yd}_{\Sigma}^{-1}(w) \mid \varphi(\zeta) \cdot \rho(\mathrm{rt}(\zeta)) \neq 0\}$ is also finite. Hence $\mathcal{L}$ is plain. Moreover, we have

$$\lambda_{\mathcal{G}}(w) = \sum_{\zeta \in D_{\mathcal{G}}(w), \mathrm{wt}(\zeta) \neq 0} \mathrm{wt}(\zeta) = \sum_{\zeta \in \mathrm{yd}_{\bar{P}}^{-1}(w)} \varphi(\zeta) \cdot \rho(\mathrm{rt}(\zeta)) = \sum_{\zeta \in \mathrm{yd}_{\bar{P}}^{-1}(w)} \|\mathcal{L}\|(\zeta) = \mathrm{yd}(\|\mathcal{L}\|)(w),$$

where second equality follows using that $D_{\mathcal{G}}(w) \subseteq \mathrm{yd}_{\bar{P}}^{-1}(w)$ and the note made on the values of $\varphi(\zeta) \cdot \rho(\mathrm{rt}(\zeta))$ for trees not in $D_{\mathcal{G}}(w)$. □

**Lemma 4.** $\mathrm{CFL}_{\mathrm{p}}(K) \subseteq \mathrm{yd}(\text{bud-Rec}_{\mathrm{p}}(K) \cap \text{tdd-Rec}_{\mathrm{p}}(K))$.

*Proof.* Let $\mathcal{G} = (N, X, Z, P, \mathrm{wt})$ be a plain CF($K$)-grammar. We define the $K$-wta $\mathcal{A} = (Q, \bar{P}, X, \delta, \kappa)$, where

- $Q = N \cup \{\overline{x} \mid x \in X\}$,
- $\bar{P}$ is the ranked alphabet defined in Section 2.2,
- the family $\delta$ is defined as follows:

– for every $k \geq 1$, rule $r = (A \to \alpha_1 \dots \alpha_k) \in \bar{P}_k$ with $\alpha_1, \dots, \alpha_k \in (N \cup X)$, and $q_1, \dots, q_k, q \in Q$, we let $\delta_k(q_1 \dots q_k, r, q) = \mathrm{wt}(r)$ if $q = A$ and

$$q_i = \begin{cases} \alpha_i & \text{if } \alpha_i \in N \\ \overline{x} & \text{if } \alpha_i = x \in X, \end{cases}$$

for $1 \leq i \leq k$; and we let $\delta_k(q_1 \dots q_k, r, q) = 0$ for every other choice of $q_1, \dots, q_k$ and $q$,

– for every $r = (A \to \varepsilon) \in \bar{P}_0$ and $q \in Q$, we define $\delta_0(r, q) = \mathrm{wt}(r)$ if $q = A$ and $\delta_0(r, q) = 0$ otherwise,

– for every $x \in X$ and $q \in Q$, we define $\delta_0(x, q) = 1$ if $q = \overline{x}$ and $\delta_0(x, q) = 0$ otherwise,

• for every $q \in Q$, $\kappa(q) = 1$ if $q = Z$ and $\kappa(q) = 0$ otherwise.

It is obvious that $\mathcal{A}$ is both bu-deterministic and td-deterministic. We will show that it is also plain and that $\lambda_{\mathcal{G}} = \mathrm{yd}(\|\mathcal{A}\|)$.

For every $\zeta \in T_{\bar{P}}(X)$, there is a distinguished run $\omega_\zeta \in R_{\mathcal{A}}(\zeta)$ defined for each $p \in \mathrm{pos}(\zeta)$ by

$$\omega_\zeta(p) = \begin{cases} \mathrm{lhs}(r) & \text{if } \zeta(p) = r \text{ for some } r \in \bar{P}, \\ \overline{x} & \text{if } \zeta(p) = x \text{ for some } x \in X. \end{cases}$$

The transition mappings of $\mathcal{A}$ are designed in such a way that, for every $\zeta \in T_{\bar{P}}(X)$ and $\omega \in R_{\mathcal{A}}(\zeta)$, we have $\delta^*(\omega) = 0$ if $\omega \neq \omega_\zeta$, and

$$\delta^*(\omega_\zeta) \cdot \kappa(\omega_\zeta(\varepsilon)) = \begin{cases} \mathrm{wt}(\zeta) & \text{if } \zeta \in D_{\mathcal{G}}(w) \text{ for some } w \in X^*, \\ 0 & \text{otherwise.} \end{cases}$$

This, the fact that $\mathcal{G}$ is plain, and that $D_{\mathcal{G}}(w) \subseteq \mathrm{yd}_{\bar{P}}^{-1}(w)$ implies that if $K$ is not complete, then the set $\{\zeta \in \mathrm{yd}_{\bar{P}}^{-1}(w) \mid \delta^*(\omega_\zeta) \cdot \kappa(\omega_\zeta(\varepsilon)) \neq 0\}$ is finite for every $w \in X^*$. This means that $\mathcal{A}$ is plain. Furthermore, for every $w \in X^*$, we have

$$\lambda_{\mathcal{G}}(w) = \sum_{\zeta \in D_{\mathcal{G}}(w), \mathrm{wt}(\zeta) \neq 0} \mathrm{wt}(\zeta) = \sum_{\zeta \in \mathrm{yd}_{\bar{P}}^{-1}(w)} \delta^*(\omega_\zeta) \cdot \kappa(\omega_\zeta(\varepsilon)) =$$

$$\sum_{\zeta \in \mathrm{yd}_{\bar{P}}^{-1}(w)} \sum_{\omega \in R_{\mathcal{A}}(\zeta)} \delta^*(\omega) \cdot \kappa(\omega(\varepsilon)) = \sum_{\zeta \in \mathrm{yd}_{\bar{P}}^{-1}(w)} \|\mathcal{A}\|(\zeta) = \mathrm{yd}(\|\mathcal{A}\|)(w),$$

where $\omega_\zeta$ is the particular run in $R_{\mathcal{A}}(\zeta)$ defined above. The second equality holds because $D_{\mathcal{G}}(w) \subseteq \mathrm{yd}_{\bar{P}}^{-1}(w)$ and the note made on $\delta^*(\omega_\zeta)$. The third one holds because $\delta^*(\omega) = 0$ for $\omega \neq \omega_\zeta$. □

To prove that $\mathrm{yd}(\mathrm{Rec}_{\mathrm{p}}(K)) \subseteq \mathrm{CFL}_{\mathrm{p}}(K)$ we need the following preparation. A $K$-wta $\mathcal{A} = (Q, \Sigma, X, \delta, \kappa)$ has *Boolean root weights* (see [13, Sec. 3.2]) if $\kappa(q) \in \{0, 1\}$ for every $q \in Q$. In this case we replace $\kappa$ by the set $F = \{q \in Q \mid \kappa(q) = 1\}$

and write $\mathcal{A} = (Q, \Sigma, X, \delta, F)$. For a $\xi \in T_\Sigma(X)$, let $R^F_\mathcal{A}(\xi) = \{\omega \in R_\mathcal{A}(\xi) \mid \omega(\varepsilon) \in F\}$. Then it is easy to see that $\|\mathcal{A}\|(\xi) = \sum_{\omega \in R^F_\mathcal{A}(\xi)} \delta^*(\omega)$. Moreover,

$$U_\mathcal{A}(w) = \{\xi \in \mathrm{yd}_\Sigma^{-1}(w) \mid \exists(\omega \in R^F_\mathcal{A}(\xi)) : \delta^*(\omega) \neq 0\}$$

and

$$\lambda_\mathcal{A}(w) = \sum_{\xi \in U_\mathcal{A}(w), \omega \in R^F_\mathcal{A}(\xi)} \delta^*(\omega).$$

In [4, Thm. 6.1.6] it is shown that $K$-wta and $K$-wta with Boolean root weights are equally powerful (see [13, Thm. 3.6]). We will now give another, slightly modified proof.

**Lemma 5.** *For each $K$-wta $\mathcal{A}$ there is a $K$-wta $\mathcal{A}'$ with Boolean root weights such that $\|\mathcal{A}\| = \|\mathcal{A}'\|$ and $U_\mathcal{A}(w) = U_{\mathcal{A}'}(w)$ for every $w \in \Sigma^*$.*

*Proof.* Let $\mathcal{A} = (Q, \Sigma, X, \delta, \kappa)$ be a $K$-wta. We construct a $K$-wta $\mathcal{A}'$ with Boolean root weights such that $\|\mathcal{A}\| = \|\mathcal{A}'\|$. First, let $F = \{q_f \mid q \in Q\}$ be a disjoint copy of $Q$ and let $Q' = Q \cup F$. Then construct $\mathcal{A}' = (Q', \Sigma, X, \delta', F)$, where $\delta'$ is defined as follows:

- for every $y \in (\Sigma_0 \cup X)$ and $q \in Q$, let

$$\delta'_0(y, q) = \delta_0(y, q) \text{ and } \delta'_0(y, q_f) = \delta_0(y, q) \cdot \kappa(q), \text{ and}$$

- for every $k \geq 1$, $\sigma \in \Sigma_k$, $q_1, \ldots, q_k \in Q'$, and $q \in Q$, let

$$\delta'_k(q_1 \ldots q_k, \sigma, q) = \begin{cases} \delta_k(q_1 \ldots q_k, \sigma, q) & \text{if } q_1, \ldots, q_k \in Q \\ 0 & \text{otherwise} \end{cases}$$

and

$$\delta'_k(q_1 \ldots q_k, \sigma, q_f) = \begin{cases} \delta_k(q_1 \ldots q_k, \sigma, q) \cdot \kappa(q) & \text{if } q_1, \ldots, q_k \in Q \\ 0 & \text{otherwise.} \end{cases}$$

Now we will explore the relation between the runs of $\mathcal{A}$ and of $\mathcal{A}'$ on a tree $\xi \in T_\Sigma(X)$. First, we note that $R_\mathcal{A}(\xi) \subseteq R_{\mathcal{A}'}(\xi)$ because $Q \subseteq Q'$. Actually, a run $\omega \in R_{\mathcal{A}'}(\xi)$ is in $R_\mathcal{A}(\xi)$ if and only if $\omega(p) \in Q$ for every $p \in \mathrm{pos}(\xi)$. Next, we introduce the notation

$$\widehat{R}^F_{\mathcal{A}'}(\xi) = \{\omega \in R^F_{\mathcal{A}'}(\xi) \mid \omega(p) \in Q \text{ for every } p \in \mathrm{pos}(\xi) \text{ with } p \neq \varepsilon\}.$$

Note that, for each $\omega \in \widehat{R}^F_{\mathcal{A}'}(\xi)$, we have $\omega(\varepsilon) = q_f$ for some $q \in Q$. Moreover, there is a bijection from $\widehat{R}^F_{\mathcal{A}'}(\xi)$ to $R_\mathcal{A}(\xi)$ defined by the correspondence $\omega \mapsto \widehat{\omega}$, where $\widehat{\omega}(\varepsilon) = q$ if $\omega(\varepsilon) = q_f$ and $\widehat{\omega}(p) = \omega(p)$ for any other $p \in \mathrm{pos}(\xi)$ with $p \neq \varepsilon$.

It follows from the construction that, for every $\xi \in T_\Sigma(X)$ and $\omega \in R_{\mathcal{A}'}(\xi)$, we have

$$\delta'^*(\omega) = \begin{cases} \delta^*(\omega) & \text{if } \omega \in R_{\mathcal{A}}(\xi) \\ \delta^*(\widehat{\omega}) \cdot \kappa(\widehat{\omega}(\varepsilon)) & \text{if } \omega \in \widehat{R}^F_{\mathcal{A}'}(\xi) \\ 0 & \text{otherwise,} \end{cases}$$

where $\omega \mapsto \widehat{\omega}$ is the bijection defined above.

Then we find that

$$\|\mathcal{A}'\|(\xi) = \sum_{\omega \in R^F_{\mathcal{A}'}(\xi)} \delta'^*(\omega) = \sum_{\omega \in \widehat{R}^F_{\mathcal{A}'}(\xi)} \delta'^*(\omega) = \sum_{\omega \in R_{\mathcal{A}}(\xi)} \delta^*(\omega) \cdot \kappa(\omega(\varepsilon)) = \|\mathcal{A}\|(\xi)$$

holds for every $\xi \in T_\Sigma(X)$. The second equality holds because $\delta'^*(\omega) = 0$ for each $\omega \in (R^F_{\mathcal{A}'}(\xi) \setminus \widehat{R}^F_{\mathcal{A}'}(\xi))$ and the third one holds by the bijection between $\widehat{R}^F_{\mathcal{A}'}(\xi)$ and $R_{\mathcal{A}}(\xi)$ described above. This proves that $\|\mathcal{A}'\| = \|\mathcal{A}\|$.

Now, let $w \in X^*$. To see that $U_{\mathcal{A}}(w) = U_{\mathcal{A}'}(w)$, first we note that

$$U_{\mathcal{A}'}(w) = \{\xi \in \mathrm{yd}^{-1}_\Sigma(w) \mid \exists(\omega \in \widehat{R}^F_{\mathcal{A}'}(\xi)) : \delta'^*(\omega) \neq 0\}.$$

Due to the bijection between $\widehat{R}^F_{\mathcal{A}'}(\xi)$ and $R_{\mathcal{A}}(\xi)$ for every $\xi \in T_\Sigma(X)$, we have $U_{\mathcal{A}'}(w) = U_{\mathcal{A}}(w)$. □

**Corollary 2.** *For each plain $K$-wta $\mathcal{A}$ there is a plain $K$-wta $\mathcal{A}'$ with Boolean root weights such that $\lambda_{\mathcal{A}} = \lambda_{\mathcal{A}'}$.*

*Proof.* Let $\mathcal{A}$ be a plain $K$-wta and construct $\mathcal{A}'$ as in Lemma 5. Since $U_{\mathcal{A}}(w) = U_{\mathcal{A}'}(w)$ for every $w \in \Sigma^*$, it follows that $\mathcal{A}'$ is also plain. Furthermore, since $\|\mathcal{A}\| = \|\mathcal{A}\|'$, we have

$$\lambda_{\mathcal{A}} = \mathrm{yd}(\|\mathcal{A}\|) = \mathrm{yd}(\|\mathcal{A}'\|) = \lambda_{\mathcal{A}'}.$$

□

**Lemma 6.** $\mathrm{yd}(\mathrm{Rec}_{\mathrm{p}}(K)) \subseteq \mathrm{CFL}_{\mathrm{p}}(K)$.

*Proof.* Let $\mathcal{A} = (Q, \Sigma, X, \delta, F)$ be a plain $K$-wta with Boolean root weights (by Corollary 2 without loss of generality). We construct a plain $\mathrm{CF}(K)$-grammar $\mathcal{G}$ such that $\lambda_{\mathcal{A}} = \lambda_{\mathcal{G}}$. Let $\mathcal{G} = (N, X, Z, P, \mathrm{wt})$, where

- $N = \{Z\} \cup \big(Q \times (\Sigma \cup X)\big)$, where $Z$ is a new symbol,
- $P$ and wt are defined as follows:
  - for every $(q, y) \in F \times (\Sigma \cup X)$, the rule $r = (Z \to (q, y))$ is in $P$ with $\mathrm{wt}(r) = 1$,
  - for every $k \geq 1$, $(q, \sigma) \in Q \times \Sigma_k$, $(q_1, y_1), \ldots, (q_k, y_k) \in Q \times (\Sigma \cup X)$, the rule $r = ((q, \sigma) \to (q_1, y_1) \ldots (q_k, y_k))$ is in $P$ with $\mathrm{wt}(r) = \delta_k(q_1 \ldots q_k, \sigma, q)$,

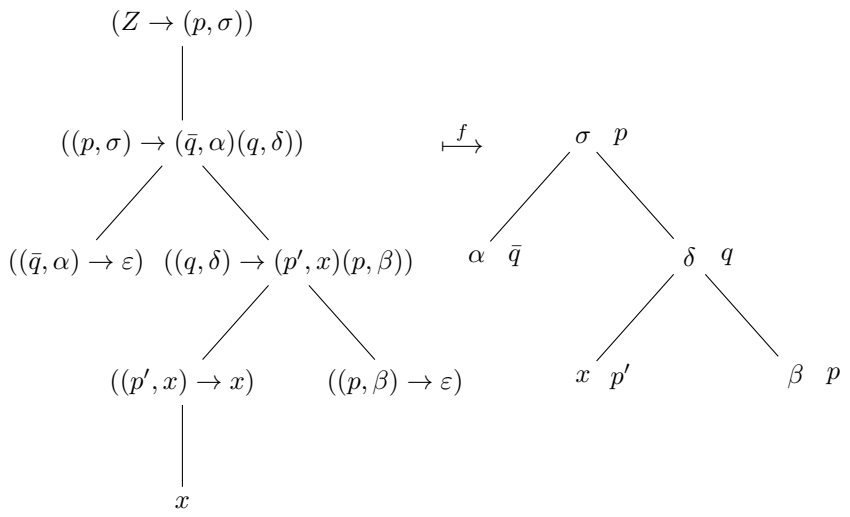



Figure 2: A visualization of the bijection $f$ given in Lemma 6.

- for every $(q, \sigma) \in Q \times \Sigma_0$, the rule $r = ((q, \sigma) \to \varepsilon)$ is in $P$ with $\mathrm{wt}(r) = \delta_0(\sigma, q)$, and
- for every $(q, x) \in Q \times X$, the rule $r = ((q, x) \to x)$ is in $P$ with $\mathrm{wt}(r) = \delta_0(x, q)$.

First we show that, for every $w \in X^*$, there is a bijection $f$ between the sets

$$D_{\mathcal{G}}(w) \text{ and } \{(\xi, \omega) \mid \xi \in \mathrm{yd}_{\Sigma}^{-1}(w), \omega \in R_{\mathcal{A}}^{F}(\xi)\}$$

such that if $f(\zeta) = (\xi, \omega)$ for $\zeta \in D_{\mathcal{G}}(w)$, then $\mathrm{wt}(\zeta) = \delta^*(\omega)$. To find such a bijection, for each tree $\zeta \in D_{\mathcal{G}}(w)$, we define $\hat{\zeta} \in T_{\Sigma}(X)$ and $\omega_{\zeta} \in R_{\mathcal{A}}^{F}(\hat{\zeta})$ as follows. Let $\mathrm{pos}(\hat{\zeta}) = \{p \in \mathrm{pos}(\zeta|_1) \mid (\zeta|_1)(p) \notin X\}$. Moreover, for every $p \in \mathrm{pos}(\hat{\zeta})$, let $\hat{\zeta}(p)$ be the second, while $\omega_{\zeta}(p)$ be the first component of $\mathrm{lhs}(r)$, where $r = (\zeta|_1)(p)$ (see Figure 2 for example). It can be seen that the mapping $f : \zeta \mapsto (\hat{\zeta}, \omega_{\zeta})$ is a bijection which satisfies the condition $\mathrm{wt}(\zeta) = \delta^*(\omega_{\zeta})$.

Now, assume that $K$ is not complete and let $w \in X^*$. Since $\mathcal{A}$ is plain and thus $U_{\mathcal{A}}(w)$ is finite, the set $\{(\xi, \omega) \mid \xi \in \mathrm{yd}_{\Sigma}^{-1}(w), \omega \in R_{\mathcal{A}}^{F}(\xi), \delta^*(\omega) \neq 0\}$ is also finite because, for every $\xi \in \mathrm{yd}_{\Sigma}^{-1}(w)$, the set $R_{\mathcal{A}}^{F}(\xi)$ is also finite. Then, due to the bijection defined above, the set $\{\zeta \in D_{\mathcal{G}}(w) \mid \mathrm{wt}(\zeta) \neq 0\}$ is also finite, which proves that $\mathcal{G}$ is plain.

Finally, we show that $\lambda_{\mathcal{G}} = \lambda_{\mathcal{A}}$. Indeed, for every $w \in X^*$,

$$\lambda_{\mathcal{G}}(w) = \sum_{\zeta \in D_{\mathcal{G}}(w), \mathrm{wt}(\zeta) \neq 0} \mathrm{wt}(\zeta) = \sum_{\substack{\xi \in \mathrm{yd}_{\Sigma}^{-1}(w) \\ \omega \in R_{\mathcal{A}}^{F}(\xi), \delta^*(\omega) \neq 0}} \delta^*(\omega) =$$

$$\sum_{\xi \in U_{\mathcal{A}}(w), \omega \in R_{\mathcal{A}}^{F}(\xi)} \delta^*(\omega) = \lambda_{\mathcal{A}}(w),$$

where the second equality holds due to the bijection $f$ defined above, the third one holds because we extend the sum with finitely many 0, and the fourth one holds from the definition of $\lambda_{\mathcal{A}}$ and Lemma 1(1). □

# References


[1] A. Alexandrakis and S. Bozapalidis, Weighted grammars and Kleene's theorem. *Information Processing Letters*, 24(1):1–4, January 1987.

[2] J. Berstel and C. Reutenauer, Recognizable formal power series on trees. *Theoret. Comput. Sci.*, 18(2):115–148, 1982.

[3] B. Borchardt, A Pumping Lemma and Decidability Problems for Recognizable Tree Series. *Acta Cyb.*, 16(4):509–544, 2004.

[4] B. Borchardt, The Theory of recognizable Tree Series. Verlag für Wissenschaft und Forschung, Berlin, 2005 (PhD thesis, TU Dresden, Germany, 2004).

[5] N. Chomsky and M.P. Schützenberger, The algebraic theory of context-free languages. In: Computer Programming and Formal Systems (eds. P. Braffort and D. Hirschberg), North-Holland, 118–161, North-Holland, 1963.

[6] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi, Tree Automata Techniques and Applications. Available on: http://www.grappa.univ-lille3.fr/tata, 1997.

[7] J. Doner, Tree acceptors and some of their applications. *J. Comput. System Sci.*, 4 (1970) 406–451.

[8] M. Droste and H. Vogler, The Chomsky-Schützenberger theorem for quantitative context-free languages. *Int. J. of Foundations of Comput. Sci.*, 25(8):955–969, 2014.

[9] S. Eilenberg, Automata, Languages, and Machines – Volume A. Academic Press, 1974, Pure and Applied Mathematics, 59.

[10] J. Engelfriet, Context–free grammars with storage. Technical Report 86-11, University of Leiden, 1986. See also: arXiv:1408.0683 [cs.FL], 2014.

[11] Z. Ésik and W. Kuich, Formal tree series. *J. of Automata, Languages, and Combinatorics*, 8(2):219–285, 2003.

[12] Z. Fülöp, Local Weighted Tree Languages. *Acta Cybernetica*, 22 (2015) 393-402.

[13] Z. Fülöp and H. Vogler, Weighted tree automata and tree transducers. *in: Handbook of Weighted Automata (eds. M. Droste, W. Kuich, and H. Vogler)*, chapter 9, 313–403, Springer-Verlag, 2009.

[14] F. Gécseg and M. Steinby, *Tree Automata*. Akadémiai Kiadó, 1984. See also: arXiv:1509.06233v1 [cs.FL] 21 Sep 2015.

[15] F. Gécseg and M. Steinby, Tree languages. *in: Handbook of Formal Languages (eds. G. Rozenberg and A. Salomaa)*, Vol. 1, pp.1-68, Springer-Verlag, 1997.

[16] J.A. Goguen, J.W. Thatcher, E.G. Wagner, and J.B. Wright, Initial algebra semantics and continuous algebras. *J. ACM*, 24:68–95, 1977.

[17] J.S. Golan, *Semirings and their Applications*. Kluwer Academic Publishers, Dordrecht, 1999.

[18] U. Hebisch and H.J. Weinert, *Semirings - Algebraic Theory and Applications in Computer Science*. World Scientific, Singapore, 1998.

[19] W. Kuich and A. Salomaa, *Semirings, Automata, Languages*. Monogr. Theoret. Comput. Sci. EATCS Series, Volume 5, Springer-Verlag, 1986.

[20] M. Magidor and G. Moran, Finite Automata over Finite Trees. Technical report 30, Hebrew University, Jerusalem, 1969.

[21] I. Petre and A. Salomaa, Algebraic systems and pushdown automata. *In: Handbook of Weighted Automata (eds. M. Droste, W. Kuich, and H. Vogler)*, chapter 7, 257–289, Springer-Verlag, 2009.

[22] W.C. Rounds, Mappings and grammars on trees. *Math. Systems Theory* 4 (1970) 257–287.

[23] W.C. Rounds, Tree-oriented proofs of some theorems on context-free and indexed languages. 2nd Ann. Assoc. Comput. Sci. Symp. on Theory of Computing, 109–116, 1970.

[24] A. Salomaa and M. Soittola, *Automata-Theoretic Aspects of Formal Power Series*. Texts and Monographs in Computer Science, Springer-Verlag, 1978.

[25] M. Steinby, Algebras as tree automata. Record Coll. Universal Algebra, Szeged 1975, North-Holland, 441–455, 1977.

[26] J.W. Thatcher, Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *J. Comput. Syst. Sci.*, 1 (1967) 317-322.

[27] J.W. Thatcher, Generalized$^2$ sequential machine maps. *J. Comput. Syst. Sci.*, 4 (1970) 339-367.

[28] J.W. Thatcher, Tree automata: an informal survey. *in Currents in the Theory of Computing (ed. A. V. Aho)*, 143–172, Prentice Hall, 1973.

*Received 24th May 2018*

Acta Cybernetica **23** (2018) 887–902.

# DFS is Unsparsable and Lookahead Can Help in Maximal Matching*

Kitti Gelle[a] and Szabolcs Iván[a]

**Abstract**

In this paper we study two problems in the context of fully dynamic graph algorithms that is, when we have to handle updates (insertions and removals of edges), and answer queries regarding the current graph, preferably with a better time bound than that when running a classical algorithm from scratch each time a query arrives. In the first part we show that there are dense (directed) graphs having no nontrivial strong certificates for maintaining a depth-first search tree, hence the so-called sparsification technique cannot be applied effectively to this problem. In the second part, we show that a maximal matching can be maintained in an (undirected) graph with a deterministic amortized update cost of $O(\log m)$ (where $m$ is the all-time maximum number of the edges), provided that a lookahead of length $m$ is available, i.e. we can "take a peek" at the next $m$ update operations in advance.

**Keywords:** dynamic graphs, depth-first search, sparse strong certificate, maximal matching, lookahead

## 1 Introduction and notation

In the past two decades, there has been a growing interest in developing a framework of algorithm design for *dynamic graphs*, that is, graphs which are subject to *updates* – in our case, additions and removals of an edge at a time. The aim of a so-called fully dynamic algorithm (here "fully" means that both addition and removal are permitted) is to maintain the result of the algorithm after each and every update of the graph, in a time bound significantly better than recomputing it from scratch each time.

While there are plenty of ad hoc algorithms for specific problems (see e.g. [1, 3, 4, 5, 7, 8]), there are also some generic methods, one of them being the *sparsificition* technique developed in [6]. This technique can speed up the computation of the query in question, achieving the same time complexity as if the query were run on

*Work of Szabolcs Iván was supported by NKFI grant no. 108448. Work of Kitti Gelle was supported by the ÚNKP-17-3 New National Excellence Program of the Ministry of Human Capacities.

[a]University of Szeged, Hungary, E-mail: {kgelle,szabivan}@inf.u-szeged.hu

DOI: 10.14232/actacyb.23.3.2018.10

a sparse graph. In order for sparsification to be applicable, it is necessary for the problem to have *sparse strong certificates*. Essentially, such a certificate for a graph $G$ is a sparse graph $G'$ on which the query should produce the same output. In [11], several graph problems were shown *not* to have a sparse strong certificate, so the technique cannot be applied to these problems (we call such properties *unsparsable*). The authors of [11] left open the question whether the *depth-first search* problem (that is, given a graph $G$ and a vertex $v$ of $G$, construct a depth-first search tree of $G$ from $v$ as a root) has sparse strong certificates or not. One of the results of the current paper is that this is not the case: there are dense graphs having no nontrivial certificate at all for this property, thus sparsification cannot be used to speed up the computation of a depth-first search tree in a dynamic graph. Although our method is still an ad hoc construction, we do hope that it can give a better insight on the nature of problems having sparse strong certificates (like edge and vertex connectivity, bipartiteness and minimum spanning tree, to name but a few).

Also in [11], a systematic investigation of dynamic graph problems in the presence of a so-called *lookahead* was initiated: although the stream of update operations can be arbitrarily large and possibly builds up during the computation time, in actual real-time systems it is indeed possible to have some form of *lookahead* available. That is, the algorithm is provided with some prefix of the update sequence of some length (for example, in [11] an assembly planning problem is studied in which the algorithm can access the prefix of the sequence of future operations to be handled of length $\Theta(\sqrt{m/n}\log n))$, where $m$ and $n$ are the number of edges and nodes, respectively. Similarly to the results of [11] (where the authors devised dynamic algorithms using lookahead for the problems of strongly connectedness and transitive closure), we will execute the tasks in batches: by looking ahead at $t = O(m)$ future update operations, we treat them as a single batch, preprocess our current graph based on the information we get from the complete batch, then we run all the updates, one at a time, on the appropriately preprocessed graph. This way, we achieve an amortized update cost of $O(\log m)$ for maintaining a maximal matching.

In this paper, a graph $G$ is viewed as a set (or list) of edges, with $|G|$ standing for its cardinality. This way notions like $G \cup H$ for two graphs $G$ and $H$ (sharing the common set $V(G) = V(H)$ of vertices) are well-defined.

**Related work.** There is an interest in computing a maximum (i.e. maximum cardinality) or maximal (i.e. non-expandable) matching in the fully dynamic setting. There is no "best-so-far" algorithm, since the settings differ: Baswana, Gupta and Sen [2] presented a randomized algorithm for maximal matching, having an $O(\log n)$ expected amortized time per update. (Note that algorithms for maximal matching automatically provide 2-approximations for maximum matching and also vertex cover.) For the deterministic variant, Ivkovič and Lloyd [9] defined an algorithm with an $O((n+m)^{0.7072})$ amortized update time, which was improved to an amortized $O(\sqrt{m})$ update cost by Neiman and Solomon [13]. For maximum matching, Onak and Rubinfeld [14] developed a randomized algorithm that achieves a $c$-approximation for some constant $c$, with an $O(\log^2 n)$ expected amortized update time. To maintain an exact maximum cardinality matching, Micali and Vazirani

[12] gave an algorithm with a worst-case update time of $O(\sqrt{n} \cdot m)$. Allowing randomization, an update cost of $O(n^{1.495})$ is achievable due to Sankowski [15].

We are not aware of any results on allowing lookahead for any of the matching problems, but the notion has been applied to several problems in this field: following the seminal work of Khanna, Motwani and Wilson [11], where lookahead was investigated for the problems of maintaining the transitive closure and the strongly connectedness of a directed graph, Sankowski and Mucha [16] also considered the transitive closure with lookahead via the dynamic matrix inverse problem, devising a randomized algorithm, and Kavitha [10] studied the dynamic matrix rank problem.

# 2 Depth-first search trees

One of the main results of the current paper is that the (general) depth-first search tree property DFS is also unsparsable. Although this is again carried out in an ad hoc way, we hope that it might give an insight into the structure of unsparsable properties.

## 2.1 Notation

We use the following notions introduced in [11] in this form.

A *graph property* is an arbitrary function $\mathcal{P}$ which maps graphs to *nonempty sets* of objects. For example, the depth-first search function DFS maps a given directed graph to several possible depth-first search forests.

The so-called sparsification technique (introduced originally in [11, 6] as a tool for studying properties of dynamic graphs) is based on the notion of *certificates*:

**Definition 1** (Strong Certificate)**.** *For a graph property $\mathcal{P}$, a* strong $\mathcal{P}$-certificate *of a graph $G$ is a graph $G'$ on the same vertex set as $G$ such that*

$$\mathcal{P}(G' \cup H) \subseteq \mathcal{P}(G \cup H)$$

*holds for any graph $H$.*

Evidently, any graph $G$ is a certificate of itself, and the following properties of *transitivity* and *monotonicity* hold [11, 6]:

- If $G'$ is a strong $\mathcal{P}$-certificate for $G$ and $G''$ is a strong $\mathcal{P}$-certificate for $G'$, then $G''$ is a strong $\mathcal{P}$-certificate for $G$ as well.
- If $G'$ and $H'$ are strong $\mathcal{P}$-certificates for $G$ and $H$, respectively, then $G' \cup H'$ is a strong $\mathcal{P}$-certificate for $G \cup H$.

In the state-of-the-art for dynamic graph algorithms, the sparsification technique can be used to develop a dynamic algorithm for a specific problem if there are *sparse* strong certificates for the given problem:

**Definition 2** (Sparse strong certificate)**.** *A property $\mathcal{P}$ has* sparse strong certificates *if, for every graph $G$ having $n$ vertices, there exists a strong $\mathcal{P}$-certificate for $G$ with $O(n)$ edges.*

If some property $\mathcal{P}$ has sparse strong certificates, having $c{\cdot}n$ edges, say, and there is a fully dynamic graph algorithm with a runtime of $T(n)$ on *sparse* graphs having $n$ nodes and $c{\cdot}n$ edges, moreover, there is an algorithm which can compute a sparse strong certificate of a graph having $n$ nodes and $2c{\cdot}n$ edges with a runtime of $T'(n)$, then (by arranging the graph into a form of a complete binary tree of its specific subgraphs) one can construct a fully dynamic algorithm solving $\mathcal{P}$ for arbitrary graphs with a runtime of $O(\log(n)(T(n)+T'(n))$. For example, if there were sparse strong certificates for DFS that are computable in $T'(n) = O(n)$ time, then the technique would yield a dynamic algorithm having an update cost of $O(n \log n)$ (note that for dense graphs this cost would improve over the naïve approach, which has an update cost of $O(m)$).

In particular, if one shows that for a given property $\mathcal{P}$, there exist graphs for arbitrarily large $n$ having $\Omega(n^2)$ edges having no nontrivial certificates (we call such properties *unsparsable*), then, as a byproduct one gets that sparsification cannot be applied effectively to speed up the computation of $\mathcal{P}$ in the dynamic setting.

In [11], a number of unsparsable properties were found: the *breadth-first* search tree property, strong connectivity, the *lexicographic* depth-first search tree property, transitive closure, diameter, minimum cut and maximum matching have been shown to be unsparsable. Most of the methods developed in [11] are applicable only to *monovalued* properties, i.e., when $\mathcal{P}(G)$ is a singleton set for every graph $G$. Indeed, all these properties are monovalued but that of the breadth-first search tree property (in which case the result is the set of all possible breadth-first trees of the input graph) which have been shown to be unsparsable using an ad hoc reasoning. The authors of [11] explicitly state that "we are unable to extend this to the case of general depth-first search tree property, and that remains an interesting open question", after showing that the (monovalued) property of lexicographic depth-first search tree property is unsparsable.

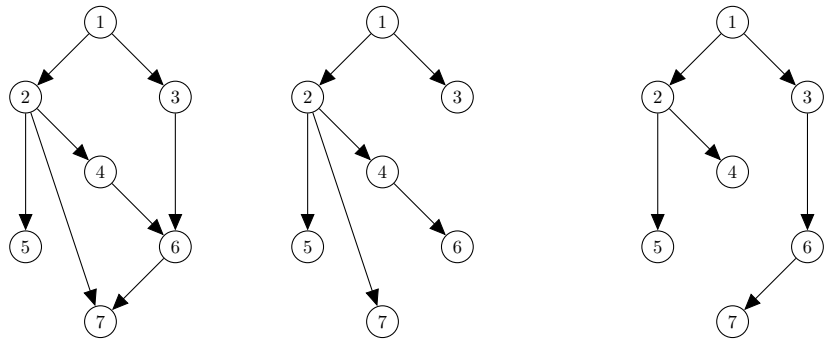
## 2.2 The property DFS is unsparsable

The property DFS assigns to a graph the set of all of its subgraphs that may be the result of a depth-first search according to *some* arbitrary ordering of the vertices, called on a specific vertex 1. For the sake of completeness, the algorithm is explicitly presented below.

```
dfs( v : Node ) {
  for each neighbour u of v do
    if( parent[u]==NULL){
      parent[u] := v
      dfs( u )
    }
}
```

```
for each node v do
  parent[v] := NULL
dfs( 1 )
```

The source of ambiguity in this algorithm is that the *order* of the neighbours of a node in which they are traversed is not specified. As an example, the reader is referred to Figure 1 where a graph $G$ and two of its possible DFS trees are depicted. Of course if there is a total ordering defined on the nodes, and the neighbours of each node are traversed according to this ordering, then the DFS tree is unique and the property becomes a *monovalued* property called *lexicographic depth-first search tree*, which, due to the results of [11], is unsparsable. Clearly, every possible depth-first search tree corresponds to a lexicographic one, e.g. the search trees depicted in Figure 1 correspond to the orderings $1 \prec 2 \prec 5 \prec 7 \prec 4 \prec 6 \prec 3$ and $1 \prec 3 \prec 6 \prec 7 \prec 2 \prec 4 \prec 5$, respectively.




(a) Original graph (b) A DFS tree from node 1 (c) Another DFS tree from node 1

Figure 1: Possible depth-first search trees

Next we show that the DFS property is unsparsable as well by means of giving an explicit family of graphs having $\Omega(n^2)$ edges such that the smallest strong certificate of any member $G$ of the family is $G$ itself. By "smallest", we mean minimal:

**Definition 3.** *A strong $\mathcal{P}$-certificate $G'$ of a graph $G$ is a* minimal *strong $\mathcal{P}$-certificate of $G$ if no proper subgraph of $G'$ is a strong $\mathcal{P}$-certificate of $G$.*

Observe that a minimal strong DFS-certificate has no edges of the form $(i, 1)$. Indeed, since 1 is always the root of any depth-first search tree, such edges cannot be tree edges and thus can be removed from a graph without changing the set of its depth-first search trees. Similarly, a minimal strong DFS-certificate does not have loop edges.

In order to show that DFS is unsparsable, we first need to prove the following three lemmas.

**Lemma 1.** *Assume $G$ is a graph such that all of its vertices are reachable from the vertex* 1*. Then the same holds in any strong* DFS*-certificate $G'$ of $G$.*

*Proof.* Clearly, any vertex $v$ is a descendant of the root node 1 in any depth-first search tree in a graph $G'$ if and only if $v$ is reachable from 1 in $G'$. Hence, if some vertex $v$ is not reachable from 1 in $G'$, then $\textsc{dfs}(G')$ consists of trees in which $v$ does not occur while $\textsc{dfs}(G)$ consists of trees in which all the nodes occur, so $\textsc{dfs}(G') \cap \textsc{dfs}(G) = \emptyset$. In particular, $\textsc{dfs}(G') \not\subseteq \textsc{dfs}(G)$ and $G'$ is not a strong DFS-certificate of $G$. □

The next lemma allows us to consider only subgraphs as possible certificates.

**Lemma 2.** *Assume $G'$ is a minimal strong* DFS*-certificate of $G$. Then $G' \subseteq G$.*

*Proof.* Assume $V(G') = V(G)$ and $G' \not\subseteq G$ is a minimal strong DFS-certificate of $G$. Then $(i, j) \in G' - G$ for some nodes $i, j \in V$. By minimality, $j \neq 1$ and $i \neq j$. There are two cases: either $i = 1$ or $i \neq 1$.

- If $i = 1$, then there exists a depth-first search tree of $G'$ in which $j$ is a depth-one node. (That is, any tree we get if we uncover $j$ first.) Since $(1, j)$ is not an edge in $G$, there is no such tree in $\textsc{dfs}(G)$ and thus $\textsc{dfs}(G') \not\subseteq \textsc{dfs}(G)$, which is a contradiction.
- If $i \neq 1$, then let $H$ be the graph consisting of the single edge $(1, i)$. It suffices to check that $\textsc{dfs}(G' \cup H) \not\subseteq \textsc{dfs}(G \cup H)$. If in $G' \cup H$ we uncover the neighbour $i$ of 1 first; then we uncover the neighbour $j$ of $i$; then we get a depth-first search tree of $G' \cup H$ in which $(i, j)$ is a tree edge. This is clearly not possible in $G \cup H$ as $(i, j)$ is not an edge in that graph. Hence $\textsc{dfs}(G' \cup H) \not\subseteq \textsc{dfs}(G \cup H)$ and $G'$ is not a strong DFS-certificate of $G$, which is a contradiction.

□

The last technical lemma of the section provides a sufficient condition for some edges being unremovable when looking for a certificate subgraph:

**Lemma 3.** *Assume $G' \subseteq G$, and $(i, j) \in G$ is an edge such that $j$ is not reachable from $i$ in $G'$; moreover, both $i$ and $j$ are reachable in $G$ from 1, and $(1, j)$ is* not *an edge in $G$.*

*Then $G'$ is* not *a strong* DFS*-certificate of $G$.*

*Proof.* Assume $G'$ is a strong DFS-certificate of $G$. By Lemma 1 we get that both $i$ and $j$ are reachable from 1 in $G'$ as well. Let $k \notin \{1, j\}$ be a node of a path from 1 to $j$ in $G'$. Notice that $k \neq i$ in this case, since by assumption $j$ is not reachable from $i$ in $G'$. Also, $k$ is thus reachable from 1 in $G'$.

Consider the graph $H$ on the same set of nodes consisting of the edges $(k, i)$, $(k, j)$, $(1, k)$ and $(1, i)$.

Then there exists a depth-first search tree of $G' \cup H$ in which $i$ and $k$ are depth-one nodes, and $j$ is a child of $k$.

To see this, first observe that neither $j$ nor $k$ is reachable from $i$ in $G' \cup H$:

- By assumption, $j$ is not reachable from $i$ in $G'$.

- Since the edges $(k, i)$ and $(1, i)$ are never used in a shortest $i \rightsquigarrow j$ path, $j$ is not reachable in $G' \cup \{(k, i), (1, i)\}$.
- Adding $(1, k)$ and $(k, j)$ does not change the transitive closure of the graph since $k$ is already reachable from 1 in $G'$, and $j$ is also reachable from $k$ in $G'$. Hence, $j$ is not reachable from $i$ in $G' \cup H$.
- Since $(k, j) \in H$, $j$ is reachable from $k$ in $G' \cup H$.
- Thus $k$ is not reachable from $i$ in $G' \cup H$.

So, if during a depth-first search on $G' \cup H$ one uncovers the node $i$ first via the edge $(1, i) \in H$, then traverses the node in an arbitrary manner, neither $j$ nor $k$ appears as the descendant of $i$ since these nodes are not reachable from $i$. Then, after finishing traversal of $i$, one uncovers $k$ via the edge $(1, k) \in H$, and then $j$ via $(k, j) \in H$. Finishing the procedure in an arbitrary way we get a depth-first search tree in which $i$ and $k$ are both depth-one nodes and $j$ is a child of $k$.

We claim that there is no such depth-first search tree of $G \cup H$, proving the above lemma. To see this, we list the possible orders in which the nodes $i, j$ and $k$ are uncovered during a depth-first search of $G \cup H$:

- Assume $i$ is uncovered first. Then, as $(i, j) \in G$, $j$ becomes a descendant of $i$ in the tree.
- Assume $j$ is uncovered first. Then $j$ cannot be a child of $k$.
- Assume $k$ is uncovered first. Then, since $(k, i) \in H$, we get that $i$ also becomes a descendant of $k$.

Hence, during any depth-first search of $G \cup H$ it cannot happen that $i$ and $k$ are both depth-one nodes and $j$ is a child of $k$, hence $\text{DFS}(G' \cup H) \not\subseteq \text{DFS}(G \cup H)$ and $G'$ is not a strong DFS-certificate of $G$. □

Now we are ready to show the main result of this section.

**Theorem 1.** *The property* DFS *is unsparsable: there exists an infinite family of graphs having $\Omega(n^2)$ edges such that for each member $G$ of the family, the only minimal strong* DFS*-certificate is $G$ itself.*

*Proof.* Let $n = 2k + 1$ be an odd number for some integer $k > 1$ and $G_n$ be the graph on $n$ vertices consisting of the following edges:

- The edges $(1, i)$ for each $2 \leq i \leq k + 1$.
- The edges $(i, j)$ for each $2 \leq i \leq k + 1$ and $k + 2 \leq j \leq 2k + 1$.

That is, a three-layered graph such that the first layer consists of the node 1, the other two layers contain $k$ nodes each, and from each node of each layer there is an edge to every node of the next layer. (See Figure 2).

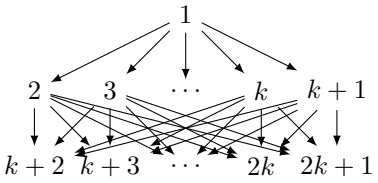



Figure 2: The graph $G_{2k+1}$

It is clear that all the nodes of $G_n$ are reachable from 1. By Lemma 2, any minimal strong DFS-certificate of $G_n$ is a subgraph $G'$ of $G_n$. Now $G'$ has to contain all the edges of the form $(1, i)$ since removing such an edge would make node $i$ unreachable from 1, contradicting Lemma 1. We claim that none of the edges $(i, j)$ with $2 \leq i \leq k+1$ and $k+2 \leq j \leq 2k+1$ can be removed. Indeed, as the removal of $(i, j)$ would make $j$ unreachable from $i$, and there is no direct edge $1 \to j$ in $G$, we see from Lemma 3 that $G'$ has to contain all the edges from the middle towards the bottom layer. Hence $G' = G_n$ is the only minimal strong DFS-certificate of $G$ and this graph has $k^2 + k = \Theta(n^2)$ edges. □

# 3 Maximal matching with lookahead

In this section we present an algorithm that maintains a maximal matching in a dynamic graph $G$ with constant query and $O(\log m)$ update time (note that $O(\log m)$ is also $O(\log n)$ as $m = O(n^2)$), provided that a *lookahead* of length $m$ is available in the sequence of (update and query) operations. This is an improvement over the currently best-known deterministic algorithm [13] that has an update cost of $O(\sqrt{m})$ without lookahead, and has the same amortized update cost as the best-known randomized algorithm [2].

In this problem, a *matching* of a(n undirected) graph $G$ is a subset $M \subseteq G$ of edges having pairwise disjoint sets of endpoints. A matching $M$ is *maximal* if there is no matching $M' \supsetneq M$ of $G$. Given a matching $M$, for each vertex $v$ of $G$ let MATE($v$) denote the unique vertex $u$ such that $(u, v) \in M$ if such a vertex exists, otherwise MATE($v$) = NULL.

In the fully dynamic version of the maximal matching problem, the update operations are edge additions $+(u, v)$, edge deletions $-(u, v)$ and the queries have the form MATE($u$).

The following is clear:

**Proposition 1.** *Suppose $G$ is a graph in which $M$ is a maximal matching. Then a maximal matching in the graph $G + (u, v)$ is*

- $M \cup \{(u, v)\}$*, if* MATE($u$) = MATE($v$) = NULL*,*
- $M$*, otherwise.*

This proposition gives the base algorithm GREEDY for computing a maximal matching in a graph:

```
Let M be an empty list of edges;
for( (u,v) ∈ G ) {
  if( MATE(u) == NULL and MATE(v) == NULL){
    MATE(u) := v; MATE(v) := u;
    insert (u,v) to M;
  }
}
return M;
```

Note that if one initializes the MATE array in the above code so that it contains some non-NULL entries, then the result of the algorithm represents a maximal matching within the subgraph of $G$ spanned by the vertices having NULL MATEs initially. Also, with $M$ represented by a linked list, the above algorithm runs in $O(m)$ total time using no lookahead. Hence, by calling this algorithm on each update operation (after inserting or removing the edge in question), we get a dynamic graph algorithm with no lookahead (hence it uses a lookahead of at most $m$ operations), a constant query cost (as it stores the MATE array explicitly) and an $O(m)$ update cost. Using this algorithm $A_1$, we build up a sequence $A_k$ of algorithms, each having a smaller update cost than the previous ones. (In a practical implementation there would be a single algorithm $A$ taking $k$ as a parameter along with the graph $G$ and the update sequence, but for proving the time complexity it is more convenient to denote the algorithms in question by $A_1$, $A_2$, and so on.)

In our algorithm descriptions the input is the current graph $G$ (which is $\emptyset$ the first time we start running the program) and a sequence $(q_1, \ldots, q_t)$ of operations. Of course as the sequence can be arbitrarily long, we do not require an explicit representation, just the access of the first $m$ elements (that is, we have a lookahead of length $m$).

**Lemma 4.** *Assume $A_k$ is a fully dynamic algorithm for maintaining a maximal matching with an $f(k) \cdot m^{1/k}$ amortized update cost, constant query cost using a lookahead of length $m$.*

*Then there is a universal constant $c$ such that there exists a fully dynamic algorithm $A_{k+1}$ that also maintains a maximal matching with $(f(k)+c(1+\log m))m^{1/k+1}$ amortized update cost, and a constant query cost using a lookahead of length $m$.*

Before proving the above lemma, we derive the main result of the section. As $A_1$ is an algorithm satisfying the conditions of this lemma with $k = 1$ and $f(k) = c_0$ for some constant $c_0$, it implies that for each $k > 1$ that there is a fully dynamic algorithm that maintains a maximal matching with an amortized update cost of $(c_0 + kc(1 + \log m))m^{1/k} = O(k \log m \cdot m^{1/k})$. Setting $k = \log m$, we get that $A_{\log m}$ has an amortized update cost of $O(\log^2 m \cdot m^{1/\log m}) = O\left(\log^2 m \cdot \left(2^{\log m}\right)^{1/\log m}\right) = O(\log^2 m \cdot 2) = O(\log^2 m)$.

Hence we get:

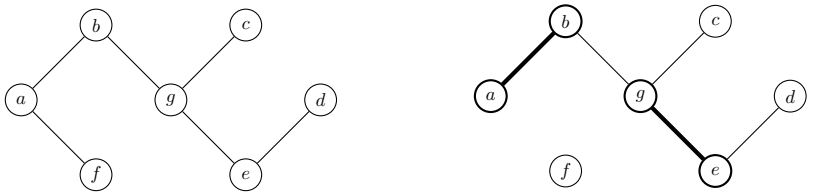
**Theorem 2.** *There exists a fully dynamic algorithm for maintaining a maximal matching with an $O(\log^2 m)$ amortized update cost and constant query cost, using a lookahead of length $m$.*

Now we prove Lemma 4 by defining the algorithm $A_{k+1}$ below.

- The algorithm $A_{k+1}$ works in *phases* and returns a graph $G$ (as an edge set) and a matching $M$ (as an edge list).
- The algorithm accesses the *global* MATE array in which the current maximal matching of the whole graph is stored. ($A_{k+1}$ might get only a subgraph of the whole actual graph as input.)
- In one phase, $A_{k+1}$ either handles a block $\vec{q} = (q_1, \ldots, q_t)$ of $t = m^{\frac{k}{k+1}}$ operations or a single operation.
- Let $G$ and $M$ respectively denote the current graph and matching we have at the beginning of a phase.
- If $|G|$ is smaller than our favorite constant 42, then the phase handles only the next operation by explicitly modifying $G$, afterwards recomputing a maximal matching from scratch, in $O(42)$ (constant) time. That is,
  1. We iterate through all the edges $(u, v) \in M$, and set MATE$[u]$ and MATE$[v]$ to NULL (in effect, we remove the "local part" $M$ of the global matching);
  2. We apply the next update operation on $G$;
  3. We set $M :=$ GREEDY$(G,$ MATE$)$.

  Otherwise the phase handles $t$ operations as follows:
  1. Using lookahead (observe that $t < m$) we collect all the edges involved in $\vec{q}$ (either by a $+(u, v)$ or a $-(u, v)$ update operation) into a graph $G'$.
  2. We construct the graph $G'' = G - G'$.
  3. We iterate through all the edges $(u, v) \in M$, and set MATE$[u] :=$ NULL, MATE$[v] :=$ NULL.
  4. We run $M :=$ GREEDY$(G'',$ MATE$)$.
  5. We call $A_k(G \cap G', (q_1, \ldots, q_t))$. Let $G^*$ and $M^*$ be the graph and matching returned by $A_k$.
  6. We set $G := G'' \cup G^*$ and $M := M \cup M^*$.

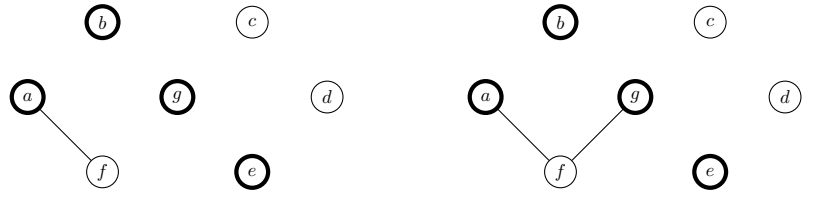
In order to give the reader a better insight, we give an example before analyzing the time complexity. To make the example more manageable, we adjust the constants as follows: we shall use the constant 1 instead of 42 (that is, if $G$ contains at most one edge, we do not make a recursive call but recompute the matching) and also, the block size $A_2$ handles in one phase will be set to 1 while $A_3$, which we call at the topmost level, will handle 3 operations in one phase.




(a) The original graph $G$. (b) $G - G'$ with a maximal matching.

Figure 3: Executing Steps $1-3$ of $A_3$ on $G$, looking ahead the operations $+(f,g)$, $-(a,f)$, $+(d,c)$

**Example 1.** Let us assume that we call the algorithm $A_3$ on the graph $G$ of Figure 3 (a). As the graph contains more edges than our threshold 1, a block of update operations of length 3 will be handled in a phase, using lookahead. (Note that if we used our actual algorithm to compute the length of the phase, we would get $6^{2/3} \approx 3.3$ as the number of edges in $G$ is $m = 6$ and in $A_3$, $k$ is 2.) Now assume the next three update operations are $+(f,g)$, $-(a,f)$ and $+(d,c)$. Thus $G' = \{(f,g),(a,f),(d,c)\}$ is the set of edges involved, that's for Step 1. In Steps 2 and 3, we construct the graph $G'' = G - G'$ and run the greedy matching algorithm on it, the (possible) result is shown in Figure 3 (b). (Note that since GREEDY does not specify the order of the edges during the traversal, the actual results can vary.) In the Figure, thick circles denote those vertices having a non-NULL mate at this point (that is, MATE$[a] = b$, MATE$[b] = a$, and so on, $c$, $d$ and $f$ having a NULL mate). Now, $A_2$ is called on $G \cap G'$ (depicted in Figure 4 (a)), and the whole block of three updates is passed to $A_2$.



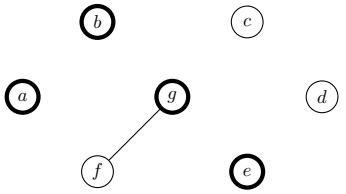

(a) The graph $G \cap G'$ (b) $A_2$ adds $(f,g)$ directly

Figure 4: Handling the first recursive call.

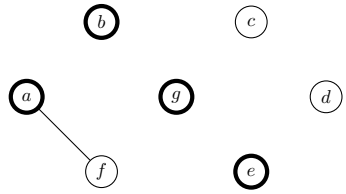
Now as the input graph of $A_2$ has only one edge, $A_2$ just handles the next update $+(f,g)$; that is, it inserts the edge $(f,g)$ into its input of Figure 4 (a) and runs GREEDY on this, resulting in the graph of Figure 4 (b). Observe that at this point MATE$[a] = b$ and MATE$[g] = e$, so neither of these two edges is added to the maximal matching managed by $A_2$. (That is, the MATE array is a global variable. This is vital: this way one can ensure that the union of the matchings of different

recursion levels is still a matching, and also ensures a constant-time query cost.)

Then, as the current graph has two edges (which is larger than the threshold), $A_2$ handles a complete block of operations in a phase. (Now the length of the block happens to be 1 so this does not make that much of a difference. Actually, as $m = 2$ and $k = 1$, the length of the block should indeed be $2^{1/2} \approx 1.4$.) Thus, using a lookahead of length 1, the only operation to be handled is $-(a, f)$. So we compute the difference graph and run GREEDY on it (Figure 5 (a)), compute the intersection graph and call $A_1$ on this along with the update sequence consisting of the single operation $-(a, f)$ (Figure 5 (b)). As the input of $A_1$ is now a graph



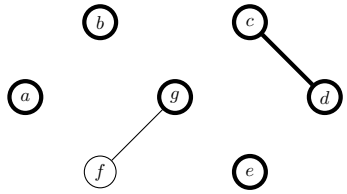

(a) The result of GREEDY run on the difference graph



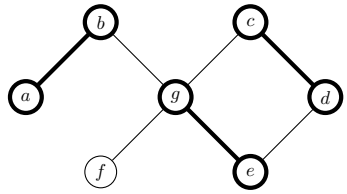

(b) The graph passed to $A_1$ along with the single update $-(a, f)$

Figure 5: Handling the second update

consisting of a single edge, it gets removed (as the edge in question is not involved in the matching, which can be seen e.g. from the MATE array, the global matching is not changed), resulting in an empty graph on which GREEDY gives an empty matching as well. Then, $A_1$ returns, as it handled the only operation it received. Now $A_2$ takes control. Concluding the second phase, it constructs the union of its intersection graph and the empty graph returned by $A_1$, so its current graph $G$ becomes the graph on Figure 5 (a). As now the graph has only one edge, the next update $+(d, c)$ is handled directly: the edge $(c, d)$ is inserted and GREEDY is run (Figure 6 (a)). Now as $A_2$ has handled its whole input block, it returns its current




(a) The edge $(c, d)$ is added to the matching by $A_2$.




(b) The current graph and matching after handling all the updates.

Figure 6: Handling the last update

graph: $A_3$ takes control and glues together its difference graph from Figure 3 (b) and the returned graph 6 (a), resulting in the graph in Figure 6 (b) which would be the starting graph of further updates. Note that in the actual algorithm, as

$2 < \log m < 3$, we would call $A_2$ at the top level and handle a block of $6^{1/2} \approx 2.44$; that is, two updates, but we deliberately chose to call $A_3$ for the sake of covering almost all the possibilities the algorithm can have (the exception being the case where a matched edge gets removed, which can be simply handled by setting the endpoints' MATE values to NULL and removing the edge from the local matching of the given recursion level).

Having completed this example, we will now show its correctness. That is, we claim that each $A_k$ maintains a maximal matching among those vertices having a NULL MATE when the algorithm is called. This is true for the greedy algorithm $A_1$. Now assuming $A_k$ satisfies our claim, let us check $A_{k+1}$. When the graph is small, the algorithm throws away its locally stored matching $M$, resetting the MATE array to its original value in the process (in fact, this is the only reason why we store the local matching at each recursion level: the global matching state can be queried by accessing the MATE array alone). Then we handle the update and run GREEDY, which is known to compute a maximal matching on the subgraph of $G$ spanned by the vertices having a NULL mate. So this case is clear.

For the second case, if a block of $t$ operations is handled, then we split the graph into two, namely a difference graph $G'$ and an intersection graph $G''$. By construction, when handling the block, the edges belonging to $G'$ do not get touched. Hence, at any time point, a maximal matching of $G$ can be computed by starting from a maximal matching of $G'$ and then extending the matching by a maximal matching in the subgraph of $G''$ not covered by the matching of $G'$. Thus, if we compute a maximal matching $M'$ in the subgraph of $G'$ spanned by the vertices having a NULL MATE, updating the MATE array accordingly (that is, calling GREEDY on $G'$), and maintaining a maximal matching $M''$ over the vertices of $G''$ having a NULL MATE after that point (which is done by $A_k$, by the induction hypothesis), we get that at any time $M' \cup M''$ is a maximal matching of $G$. Hence, the algorithm is correct.

Now we analyse the time complexity of $A_{k+1}$. When a phase handles $t$ operations, then Step 1 can be executed in $O(t \log t) = O(m \log m)$ time (if we use a self-balancing tree representation for storing our graphs, say an AVL tree). Then in Step 2, we construct the difference of the two sets of size $O(m)$ in $O(m \log m)$ time. Step 3 requires an additional time of $O(m)$, since the matching is of size $O(m)$ and it is stored as a list of edges. For Step 4, as $|G''| \leq |G| = m$, also an $O(m)$ time is required, and for Step 5, computing the intersection $G \cap G'$ requires a time of $O(m \log m)$, and $A_k$, being run on a dynamic graph having at most $t = m^{\frac{k}{k+1}}$ edges during its whole lifecycle of $t$ operations needs $t \cdot f(k) \cdot t^{1/k} = m^{\frac{k}{k+1}} \cdot f(k) \cdot m^{\frac{1}{k+1}} = f(k) \cdot m$ computation steps. Gluing together the graphs and the matchings in Step 6 needs a time of $O(m \log m) + O(m)$. Hence the total cost of Steps 1-6 handling a whole phase is $O(m) + O(m \log m) + O(m) + f(k) \cdot m + O(m \log m) + O(m) = (f(k) + c(1 + \log m))m$ for some universal constant $c$, and since a phase consists of $m^{\frac{k}{k+1}}$ operations, the amortized cost of a single operation becomes $(f(k) + c(1 + \log m))m^{\frac{1}{k+1}}$ and Lemma 4 is proved.

The careful reader may observe that a major part of the time bound comes from

the set operations. If an initialization cost of $O(n^2 \log n)$ is affordable (i.e. if there are $\Omega(n^2 \log n)$ operations in total), then we can do better:

- Each algorithm $A_k$ has an adjacency matrix as well, initialized to an all-zero matrix in the very beginning (this initialization takes the aforementioned $O(n^2 \log n)$ setup cost).
- In Step 1, edges of $G'$ are stored into this matrix (taking still $O(m)$ time).
- Now the graphs $G - G'$ and $G \cap G'$, as lists of edges, can be constructed in $O(m)$ time (since lookup in $G'$ now takes constant time instead of the previous $O(\log m)$).
- Since $G''$ is represented as an edge list, GREEDY still takes $O(m)$ time.
- After performing Step 5, we have to set the auxiliary matrix to an all-zero matrix by looking ahead once again the very same sequence and setting each accessed edge to 0. This takes $O(m)$ time.
- Also, taking the unions of the graphs and matchings upon returning can be destructive to the original lists, thus it can be done in constant time.

Hence in this case the total cost spent for a phase becomes $O((f(k) + c)m)$ for some universal constant $c$, yielding an amortized update cost of $O((k+1) \cdot m^{1/k+1})$ for $A_k$, which boils down to an amortized update cost of $O(\log m)$ by choosing $k = \log m$ and we showed:

**Theorem 3.** *There exists a fully dynamic algorithm for maintaining a maximal matching with an $O(n^2 \log n)$ initialization cost, $O(\log m)$ amortized update cost and constant query cost using a lookahead of length $m$.*

# 4 Conclusion

In this study we dealt with two problems arising in the context of fully dynamic graph algorithms. First, we showed via an ad hoc method that the depth-first search tree (or, forest) property is unsparsable; that is, there are dense graphs for this property having no nontrivial strong certificates. Thus, the technique of sparsification cannot be applied to this problem effectively – if it could be, it would result in an algorithm having in an update cost of $O(n \log n)$, but it's not. This solves an open problem mentioned in [11].

In the second, more detailed part of the study we showed that by using a *lookahead* of linear length, there is a *deterministic* algorithm achieving an $O(\log m)$ amortized update cost (after a somewhat costly initialization which, if cannot be afforded for some matter, then the update cost becomes $O(\log^2 m)$). This result shows that lookahead can help in the dynamic setting for problems other than the transitive closure (and the SCC) properties, studied in [11]: indeed, the best known

deterministic algorithm for the problem using no lookahad has an update cost of $O(\sqrt{m})$.

It is an interesting question to study further the possibilities of using lookahead for different problems, and maybe factor in also randomization as well.

# References

[1] Alberts, D. and Henzinger, M. R. Average-case analysis of dynamic graph algorithms. *Algorithmica*, 20(1):31–60, Jan 1998.

[2] Baswana, Surender, Gupta, Manoj, and Sen, Sandeep. Fully dynamic maximal matching in $O(\log n)$ update time. *SIAM Journal on Computing*, 44(1):88–113, 2015.

[3] Chan, Timothy M. Dynamic subgraph connectivity with geometric applications. *SIAM J. Comput.*, 36(3):681–694, September 2006.

[4] Demetrescu, Camil and Italiano, Giuseppe F. Trade-offs for fully dynamic transitive closure on dags: Breaking through the o(n2 barrier. *J. ACM*, 52(2):147–156, March 2005.

[5] Demetrescu, Camil and Italiano, Giuseppe F. Dynamic shortest paths and transitive closure: Algorithmic techniques and data structures. *J. Discrete Algorithms*, 4(3):353–383, 2006.

[6] Eppstein, David, Galil, Zvi, Italiano, Giuseppe F., and Nissenzweig, Amnon. Sparsification—a technique for speeding up dynamic graph algorithms. *J. ACM*, 44(5):669–696, September 1997.

[7] Henzinger, Monika R. and King, Valerie. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *J. ACM*, 46(4):502–516, July 1999.

[8] Holm, Jacob, de Lichtenberg, Kristian, and Thorup, Mikkel. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM*, 48(4):723–760, July 2001.

[9] Ivković, Zoran and Lloyd, Errol L. *Fully dynamic maintenance of vertex cover*, pages 99–111. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

[10] Kavitha, Telikepalli. Dynamic matrix rank with partial lookahead. *Theor. Comp. Sys.*, 55(1):229–249, July 2014.

[11] Khanna, S., Motwani, R., and Wilson, R. H. On certificates and lookahead in dynamic graph problems. *Algorithmica*, 21(4):377–394, Aug 1998.

[12] Micali, S. and Vazirani, V. V. An $o(\sqrt{|V|} \cdot |e|)$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*, pages 17–27, Oct 1980.

[13] Neiman, Ofer and Solomon, Shay. Simple deterministic algorithms for fully dynamic maximal matching. *ACM Trans. Algorithms*, 12(1):7:1–7:15, November 2015.

[14] Onak, Krzysztof and Rubinfeld, Ronitt. Maintaining a large matching and a small vertex cover. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 457–464, New York, NY, USA, 2010. ACM.

[15] Sankowski, Piotr. Faster dynamic matchings and vertex connectivity. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 118–126, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[16] Sankowski, Piotr and Mucha, Marcin. Fast dynamic transitive closure with lookahead. *Algorithmica*, 56(2):180–197, February 2010.

*Received 3th May 2018*

Acta Cybernetica **23** (2018) 903–919.

# Analysis of Static and Dynamic Test-to-code Traceability Information*

Tamás Gergely[a], Gergő Balogh[ab], Ferenc Horváth[a], Béla Vancsics[a], Árpád Beszédes[ac], and Tibor Gyimóthy[ab]

**Abstract**

Unit test development has some widely accepted guidelines. Two of them concern the test and code relationship, namely isolation (unit tests should examine only a single unit) and separation (they should be placed next to this unit). These guidelines are not always kept by the developers. They can however be checked by investigating the relationship between tests and the source code, which is described by test-to-code traceability links. Still, these links perhaps cannot be inferred unambiguously from the test and production code.

We developed a method that is based on the computation of traceability links for different aspects and report Structural Unit Test Smells where the traceability links for the different aspects do not match. The two aspects are the static structure of the code that reflects the intentions of the developers and testers and the dynamic coverage which reveals the actual behavior of the code during test execution.

In this study, we investigated this method on real programs. We manually checked the reported Structural Unit Test Smells to find out whether they are real violations of the unit testing rules. Furthermore, the smells were analyzed to determine their root causes and possible ways of correction.

**Keywords:** test-to-code traceability, unit testing, code coverage, test smells, refactoring

*This study was supported by the project "Integrated program for training new generation of scientists in the fields of computer science", no EFOP-3.6.3-VEKOP-16-2017-0002. The project was supported by the European Union and co-funded by the European Social Fund.

[a]Department of Software Engineering, University of Szeged, Hungary, E-mail: {gertom,geryxyz,hferenc,vancsics,beszedes,gyimothy}@inf.u-szeged.hu

[b]MTA-SZTE Research Group on Artificial Intelligence, University of Szeged, Hungary

[c]This author was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

DOI: 10.14232/actacyb.23.3.2018.11

# 1 Introduction

Unit testing is an important element of software quality assurance, and it plays an important role in software maintenance and evolution. For example, during continuous integration, unit tests are constantly re-executed and further evolved (by developers) in parallel with the system under test [13]. This is why the quality of unit tests (including maintability) is important for software quality.

There are several guidelines, design patterns, and frameworks that help the developers to write good unit test cases [17]. Among these guidelines, there are two that deal with the structural consistency between test and production code [17]. The first one is *isolation*, which means that unit tests should exercise only the unit they were designed for, while the second one is *separation*, meaning that the tests should be placed in the same logical or structural group (like packages or namespaces) as the units they are testing. These guidelines, if kept, assist both the traceability between the test and production code, and maintainability. However, some practical aspects may prevent unit test designers and developers from creating tests that completely conform to these definitions (*e.g.* calls to utility functions or general parts of the system [5, 21]). Also, refactorings and code reorganizations might detrimentally affect the fulfillment of the isolation and separation guidelines for test and production code. Places in the code where these rules are not kept can be treated as test smells ([28, 3]): they are not bugs nor do they harm maintainability by definition, but such locations should be investigated anyway.

Relations between test and production code elements may be treated as traceability links, and several approaches have been proposed for their recovery (*e.g.* [23, 22, 16, 19, 9, 7]). However, as different approaches use different information to recover traceability, these might produce different results [23].

In a previous study, we proposed a method for investigating unit test and code relationship [2]. Here, we use this method to identify so-called *Structural Unit Test Smells*, a concept which we introduce to describe structural issues in the tests with respect to the system as a whole, and not just issues in isolated pieces of code. The method uses the idea that test-to-code traceability recovered from different sources captures different type of relations. Namely, we compare traceability links recovered from static sources that reflect the intention of the test designers and from dynamic sources that reflect the actual behavior of the code during test case execution. Differences in the two types of recovered traceability might suggest test and code elements that violate isolation and separation guidelines.

In the first phase of our approach we compute the traceability links based on two fundamentally different but very basic aspects, these being (1) the static relationships of the tests and the tested code in the physical code structure, and (2) the dynamic behavior of the tests based on code coverage. In particular, we compute *clusterings* of tests and code for both static and dynamic relationships, which represent coherent sets of tests and tested code. These clusters represent sets whose elements are mutually traceable to each other, and may be beneficial over individual traceability between units and tests, which is often hard to express precisely. To compute the static clusters we use the packaging structure of the

code, while for the dynamic clustering we employ community detection [6] on the method level code coverage information.

In the next phase, these two kinds of clusterings are compared with each other. We do this using a *Cluster Similarity Graph* (CSG) that represents the computed clusters as graph nodes and connects the static and dynamic clusters that have common elements. If both approaches produce the same clusters, then they are said to agree, connecting only pairs of (one static and one dynamic) nodes in the CSG, in which case we conclude that the (test and code) elements contained in the clusters conform to the given unit testing guidelines. However, in many cases there will be discrepancies in the results obtained represented as several interconnected nodes in the CSG, which we report as Structural Unit Test Smells. There may be various reasons for these SUTSs, but they are usually some combination that violates the isolation and/or separation principles mentioned above.

To assess the practical usability of the method, in this study, we applied it on non-trivial open source Java systems and their JUnit test suites. We manually investigated the reported Structural Unit Test Smells by recovering and analysing their context and finding the root cause of the detected discrepancy between the static and dynamic traceability. We also made decisions on each SUTS as to whether it is a 'false positive' (*i.e.* the test and code conforms to unit testing rules) or whether it points to test and production code that should be reorganized in some way.

The rest of the paper is organized as follows. In the next section we provide an overview of some background information and related work, then in Section 3 we describe our traceability recovery method, with the analysis of the detected discrepancies in Section 4. In Section 5 we discuss threats to validity of the study. Lastly, in Section 6 we draw some conclusions and make some suggestions for future work.

## 2 Background and Related Work

There are different levels of testing, one of which (the lowest level) is called unit (or component) testing. Unit tests are closely related to the source code and they seek to test separate code fragments. This kind of test helps one to find implementation errors early in the coding phase, and helps to reduce the overall cost of the quality assurance activity.

Several guidelines exist that provide hints on how to write good unit tests (*e.g.* [17, 27, 20]), but there are two basic principles that are mentioned by most of them. The first is that unit tests should be isolated (*i.e.* test only the elements of the target component) and separated (*i.e.* physically or logically grouped, aligned with the unit being tested). In practice, this means that unit tests should not (even indirectly) execute production code outside the tested unit, and they should follow a clear naming and packaging convention, which reflects both the purpose of the test and structure of the given system. Several studies have examined various characteristics of the source code with which the above mentioned two aspects can

be measured and can be verified to some extent (see, for example [23]).

These two properties are necessary for the approach described in this paper. Namely, if both are strictly followed, the two automatic traceability analysis algorithms we used (one package-based and the other coverage-based), will produce the same results. However, this is not the case for realistic systems, so our approach relies on analyzing the differences between the two sets in order to infer things about the final traceability links.

Several methods have been proposed to recover traceability links between software artifacts of different types, including requirements, design documentation, code, test artifacts, and so on [24, 10]. The approaches include static and dynamic code analysis, heuristic methods, information retrieval, machine-learning, data-mining based methods.

In this study, we are concerned with a specific type of traceability, namely *test-to-code* links. The purpose of recovering such links is to assign test cases to code elements of the system under test based on the relationship that shows which code parts are tested by which tests. This information may be vital in different activities including development, testing and maintenance, as mentioned earlier.

We shall concentrate on unit tests, in which case the traceability information is mostly encoded in the source code implementing the production system and the test cases, and usually no external documentation is available for this purpose. Traceability recovery for unit test may seem straightforward at first sight, given that the basic purpose of a unit test is to test a single unit of code [4, 11]. However, in reality it is not so [16, 19].

Several studies have been conducted on this subject, which examined the problem of traceability and made suggestions about it [9, 7, 23, 22]. Most of these related studies emphasize that reliable test-to-code traceability links are difficult to obtain from a single source of information, and a combination of (or semi-automatic) methods are required. Here, we will utilize this finding to determine the test smells

Our study mainly focuses on test (and code) smell identification. The discrepancies found in the two automatic traceability analysis results can be viewed as some sort of smell, and this suggests potential problems in the structural organization of the tests and code. Code smells (first introduced by Fowler [15]) are an established concept for classifying shortcomings in the software. Similar concepts for checking software tests and test code for quality issues have also been applied. For tests that are implemented as executable code, Van Deursen *et al.* introduced the concept of *test smells*, which suggest poorly designed test code [12], and listed 11 test code smells with recommended refactorings. We can relate our study best to their concept of *Indirect Testing Smell*. Meszaros expanded the scope of the concept by describing test smells that act at a behavior or a project level, next to code-level smells [20]. Results that came after this study use these ideas in practice. For example, Breugelmans and Van Rompaey [8] present TestQ, which allows developers to visually explore test suites and quantify test smelliness. They visualized the relationship between test code and production code, and with it, engineers were able to better understand the structure and quality of the test suite of large systems [27].

Our study significantly differs from these approaches as we are not concerned with code-oriented issues in the tests, but with their dynamic behavior and relationship to their physical location in the system as a whole. We may identify *Structural Unit Test Smells* from an analysis of the discrepancies found in the automatic traceability analyses.

# 3 Method

We define *Structural Unit Test Smell* (SUTS) as those suspicious parts of either the test or production code which seem to violate best practices used during unit test creation, execution and maintenance. In this respect, they are essentially inconsistencies in the physical organization and the logical behavior of unit test code and the tested code.

## 3.1 Overview

Figure 1 provides an overview of the above process, which has several sequential phases. First, the physical organization of the *production* and *test code* into Java packages is inferred, and the required *test coverage* data is produced by executing the tests. In our setting, code coverage refers to the individual recording of all methods executed by each test case. Physical code structure and coverage will be used in the next phase as inputs to create two *clusterings* over the tests and code elements.

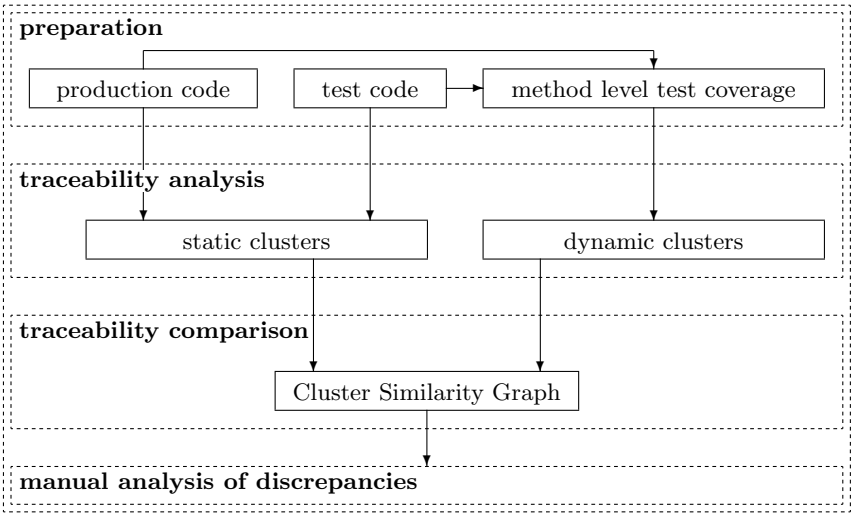



Figure 1: Overview of the method

These will represent the two types of relationships between the test and code, these being the two sets of automatically produced traceability links from two

viewpoints, namely static and dynamic. Both clusterings produce sets of clusters that are made up of a combination of tests (unit test cases) and code elements (units under test). In our case, a unit test case is a Java test method (*e.g.* using the `@Test` annotation in the JUnit framework [26]), while a unit under test is a Java production code method.

In our approach, the elements of a cluster are mutually traceable to each other, and no individual traceability is considered between individual test methods and production methods. The advantage of this is that in many cases it is impossible to uniquely assign a test case to a unit; instead *groups* of test cases and units may represent a cohesive functional unit [18]. Also, minor inconsistencies, such as helper methods that are not directly tested, are concealed in this procedure. Details about the clustering based traceability algorithms are provided in the next section.

The automatically produced traceability links of the two analyses will be compared using a helper structure called the *Cluster Similarity Graph* (CSG) [2]. This is a directed bipartite graph whose nodes (disjointly) represent the clusters of the two clusterings. Each edge of the graph connects two nodes representing one static and one dynamic cluster, and weights on them denote the level of similarity between the two corresponding cluster nodes (based on the elements contained in the two corresponding clusters). Weights can be calculated using a pairwise similarity measure. In particular, we can use the *Inclusion measure*. Let $K_1$ and $K_2$ be two clusters of different types (one static and one dynamic). The Inclusion measure $I(K_1, K_2)$ expresses to what degree the elements of $K_1$ are included in $K_2$. A value of 0 means no inclusion (fully disjoint clusters), while a value of 1 means that $K_1$ is a subset of $K_2$. Edges with a 0 inclusion value are omitted from the CSG.

Figure 2 shows an example CSG taken from one of our subject systems, `oryx`. The static clusters are represented as purple rectangles, while the dynamic clusters are represented as green boxes. The edge weights are not shown in this example. Both types of clusters contain test and code elements. Edges in the figure mean that the two connected clusters have some common items (test or production methods). For example, the elements of dynamic cluster 9 are completely contained in the static cluster `com/cloudera/oryx/common` (as dynamic cluster 9 has no common elements with other clusters), and dynamic cluster 11 shares its elements with two static ones. Note that in the example the numbers of the dynamic clusters have no special significance, while the static clusters are named after the package that contains their elements.

After we had created the CSG, we looked at it closely. It is obvious that in the ideal case static and dynamic clusterings produce the same clusters, hence the CSG contains only connected pairs of nodes. However, in practice the CSGs are not like this, and each pattern in the graph that consists of more than two connected nodes can be treated as a Structural Unit Test Smell. Therefore, we manually examined the different patterns in the CSG and tried to discover what properties of the code and tests caused them. The results for this are presented in Section 4.

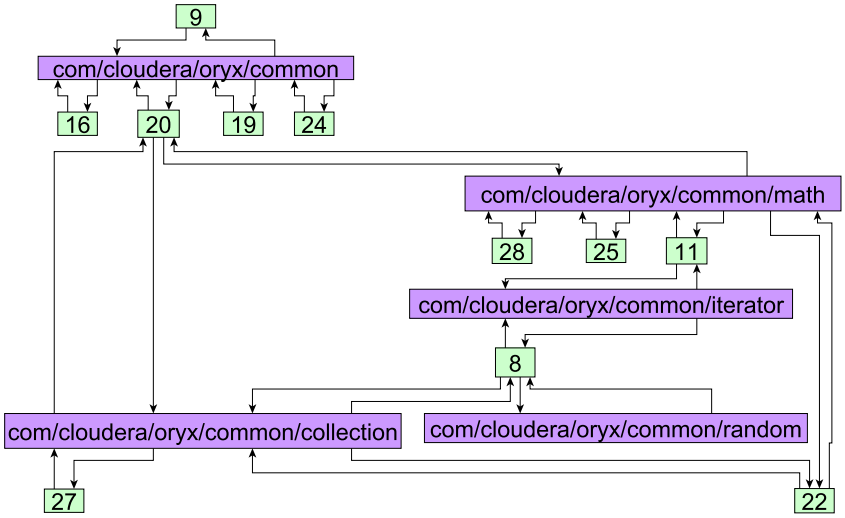



Figure 2: A part of the CSG of the `oryx` program

## 3.2 Clustering based traceability analysis

Our approach for unit test traceability recovery includes a step in which traceability links are identified automatically by analyzing the test and production code from two perspectives (static and dynamic). In both cases, clusters of code and tests are produced which jointly constitute a set of mutually traceable elements. Here, we deal with Java systems and rely on unit tests implemented in the JUnit test automation framework. In this context, elementary features are usually implemented in the production code as methods of classes, while the unit test cases are embodied as test methods. A system is then composed of methods grouped into classes and classes into packages. All of our algorithms have the method-level granularity, *i.e.* clusters are composed of production and test methods. Here, we do not explicitly take into account class information during the clusterings.

### 3.2.1 Static clustering

Through static clustering, our intention is to detect groups of tests and code that are connected together by the intention of the developer or tester. The placement of the unit tests and code elements within the package hierarchy of the system is a natural classification according to their intended role. When tests are placed within the package the tested code is located in, it helps other developers and testers to understand the connection between tests and their subjects. Hence, it is essential that the physical organization of the code and tests be reliable and reflect the developer's intentions.

Our package-based clustering simply means that we assign the fully qualified

package name of the method to each production and test method, and treat methods (of both types) belonging to the same package as members of the same cluster. Class information and higher level package hierarchy are not directly taken into account. For example, package `com.cloudera.oryx.common` and its subpackages `com.cloudera.oryx.common.math` and `com.cloudera.oryx.common.random` are treated as unique clusters containing all the methods of all the classes directly contained within them. Furthermore, we do not directly consider the physical directory and file structure of the source code elements (although in Java, these usually tell us something about the package structure).

### 3.2.2 Dynamic clustering

In order to determine the clusters of tests and code based on the actual dynamic behavior of the test suite, we apply *community detection* [6, 14] on the code coverage relations.

Code coverage in this case means that, for each test case, we record what methods were invoked during the test case execution. This forms a binary matrix called a *coverage matrix*, with test cases assigned to its rows and methods assigned to the columns. A value of 1 in a matrix cell indicates that the particular method is invoked at least once during the execution of the corresponding test case (regardless of the actual statements and paths taken within the method body), and 0 indicates that it has not been covered by the test case.

*Community detection* algorithms were originally defined on (possibly directed and weighted) graphs. Thus, in order to use the selected algorithm, we construct a graph (referred to as the *coverage graph* in the following) from the coverage matrix. The nodes are the methods and tests of the system being analyzed, and there is an edge between a method and a test node if and only if the corresponding cell in the coverage matrix is 1. There is no edge between any two nodes of the same type.

The actual algorithm we used for community detection is the Louvain Modularity method [6]. It is a greedy optimization method based on the modularity metric, which penaltizes edges between clusters and rewards edges inside clusters. The algorithm works iteratively, and each pass is composed of two phases. In the first phase it starts processing single-node clusters and continues to unify clusters until no more unification leads to an increase of modularity. In the second phase, a new graph is created by assigning a single node to each clusters of the previous graph. New edges are also computed and weighted based on the edges between the cluster elements in the previous graph. The algorithm iterates these two steps until it reaches a graph in which no nodes can be unified in terms of modularity.

## 4 Analysis of traceability discrepancies

We manually analyzed all discrepancy instances found in the results produced by the static and dynamic traceability detection approaches. In this procedure, we considered the CSGs, the associated edge weights, and examined the corresponding

parts of the production and test code. For the first step, each subject system was assigned to one of the authors of the paper for initial comprehension and analysis of the resulting patterns. The analysis required an understanding of the code structure and to some extent the intended goal of the test cases. API documentation, feature lists, and other public information were also examined during this phase. Next, the researchers made suggestions on the possible recovered traceability links and eventual code refactorings. Then, all the participants were involved in a discussion about the final decisions. The edge weights in the CSGs obtained during the analysis helped us to assess the importance of a specific cluster. For example, small inclusions were often ignored because these were in many cases due to some kind of outlier relationships that did not affect the overall structure of the clusters.

The results of the analysis were possible explanations for the reported SUTS with concrete suggestions, as well as the corresponding general guidelines for possible refactorings.

## 4.1 Subject programs

Our subject systems (see Table 1) were medium-to-large-size open-source Java programs, with unit tests implemented using the JUnit test automation framework. We chose these systems because they had a reasonable number of test cases compared to the system size.

Table 1: Subject programs

| Program | Version | LOC | Methods | Tests |
|---|---|---|---|---|
| **checkstyle** | *6.11.1* | 114K | 2 655 | 1 487 |
| **netty** | *4.0.29* | 140K | 8 230 | 3 982 |
| **orientdb** | *2.0.10* | 229K | 13 118 | 925 |
| **oryx** | *1.1.0* | 31K | 1 562 | 208 |

We modified the build processes of the systems to produce method level coverage information using the Clover coverage measurement tool [1]. For storing and manipulating the data, we used the SoDA framework [25], *e.g.* to process the coverage matrix. Then, we implemented a set of Python scripts to perform clusterings, including a native implementation of the community detection algorithm.

## 4.2 Identified *Structural Unit Test Smells*

Now, we present 8 SUTSs that we found and analyzed manually. These were the simplest smell patterns in the CSGs, where we suspected the nature of the smell and could give clear refactoring options (even if we sometimes provided more possible, contradictory options to a single smell instance). We encountered more complex patterns during our experiment (containing tens of clusters and hundreds of traceability links), but a deep analysis of these lay outside the scope of this present study.

#### 4.2.1 com/puppycrawl/tools/checkstyle/doclets

This package belongs to our subject `checkstyle` and it is composed of the class `TokenTypesDoclet` with all 5 of its methods and of the test class `TokenTypesDocletTest` with its 6 test cases. The package is used to create a configuration/property file with short descriptions of `TokenTypes` constants. There are 4 dynamic clusters connected to this static cluster. They are one for option validation (1 method, 1 test case), a cluster for file name handling and file creation (2 methods, 3 test cases), one for counting options (1 method, 1 test case) and one for static initialization (1 method, 1 test case).

**Possible explanation and refactoring:** The dynamic clusters describe the subfunctionalities correctly. This SUTS is the result of the clustering and granularity we are working with, where our units are at the package level, while we work at the method level and this enables our method to identify smaller units. Namely, the sub-features are tested separately, but the implemented (and tested) classes are not separated into different packages, which is a quite reasonable decision in this case. Hence, we treat this SUTS as a false positive, and suggest that no refactoring should be performed.

#### 4.2.2 io/netty/handler/codec/haproxy

`HAProxy` is a submodule of `netty` which is responsible for handling load balancing-related protocols. It has 7 classes, 42 methods and 30 test cases arranged in a single test class of over 1000 lines (`HAProxyMessageDecoderTest`). There are two dynamic clusters connected to it, these being one for messages and protocols (39 methods, 29 test cases) and one with two helper classes and their test case (3 methods, 1 test case).

**Possible explanation and refactoring:** A straightforward solution would be to split the package according to the dynamic behavior. However, we think that one of the resulting subpackages would be too small as a single package. Instead, the package should be divided into packages of messages, protocols and others. It would require other refactorings as well (*e.g.* involve splitting of the big test class) to produce a structure that conforms with the unit testing guidelines. However, this refactoring cannot be directly derived from the identified clusters alone. It requires a deeper analysis and knowledge of the code.

#### 4.2.3 com/cloudera/oryx/als/common

This is the core package of the `als-common` submodule of subject `oryx`. It contains comparators, custom exception classes and small utility classes, and consists of 9 classes, one interface with altogether 18 methods and 4 test classes with 17 test cases. Four dynamic clusters are connected, these being a string-to-long map utility (1 method, 6 test cases), the `DataUtils` class and related test cases (1 method, 2

test cases), another string-to-long map utility (5 methods, 2 test cases) and a set of comparators and utility methods (11 methods, 7 test cases).

**Possible explanation and refactoring:** The dynamic clusters capture the sub-functionalities correctly, and there is room for refactoring. That is, exceptions, comparators, and utilities could be separated into three different packages, one for each. From the identified dynamic clusters, we seem to have a good basis for reorganizing the code, but additional decisions and corrections are needed. This Structural Unit Test Smell turned out to be true positive.

#### 4.2.4 com/cloudera/oryx/common/io

The next *Structural Unit Test Smell* in `oryx` is the `io` package with `DelimitedDataUtils`, which is adapted from `SuperCSV` as a fast/lightweight alternative to its full API and `IOUtils`, a collection of simple utility methods related to I/O operations. It is composed of these two classes with 10 methods and the related 11 test cases. The four connected dynamic cluster are: `DelimitedDataUtils.encode` (3 methods, 5 test cases), `IOUtils.delete` (3 methods, 1 test case), `DelimitedDataUtils.decode` (2 methods, 4 test cases) and `IOUtils.copy` (2 methods, 1 test case).

**Possible explanation and refactoring:** This Structural Unit Test Smell is the result of how we define the unit in the static case. The dynamic clusters describe the sub-functionalities correctly at the method level, but the static cluster is too small to suggest some reorganization of the tests and the code. Actually, it turns out to be a false positive SUTS.

#### 4.2.5 com/cloudera/oryx/common/stats

This static package is a common submodule of `oryx` which provides different statistics, and it has 5 classes with 24 methods, and 4 test classes with 13 test cases altogether. This pattern has 4 dynamic clusters: a weighted mean implementation for floating-point weights (6 methods, 5 test cases), a similar module with integer weights (7 methods, 6 test cases), a class encapsulating a set of statistics like mean, min and max (4 methods, 1 test case) and a bean class encapsulating some characteristics of the JVM runtime environment (7 methods, 1 test case).

**Possible explanation and refactoring:** The dynamic clustering split the static package into smaller units. Following the dynamic clusters would result in four units, but here we suggest that just the bean class `JVMEnvironment` should be separated, and the others have a similar functionality. Although the SUTS is valid, the clusters can only be partially used to determine the refactoring and further knowledge is required to do it correctly.

#### 4.2.6 com/cloudera/oryx/kmeans/common

This package is the core of the `kmeans-common` submodule of `oryx` which is responsible for providing the k-means algorithm and several evaluation strategies. It includes 12 classes and 4 interfaces with 51 methods altogether, and 5 test classes about 60-70 lines long, providing 20 test cases. This case includes two coverage-based clusters (see Figure 3(a)), one for the evaluation strategies, weights, cluster centers, validity and statistics (40 methods, 17 test cases) and one that includes other evaluation strategies (11 methods, 3 test cases).

**Possible explanation and refactoring:** This package could be safely splits into two packages according to the dynamic clusters, one being responsible for the *statistical validation* and the other being responsible for the *strategies*. In Figure 3(a), cluster 32 should correspond to the *validation* and 33 should correspond to the *strategies*. This is a clear example of the situation where the Structural Unit Test Smell is not only valid, but the clusters involved in it also clearly show how the refactoring should be carried out.

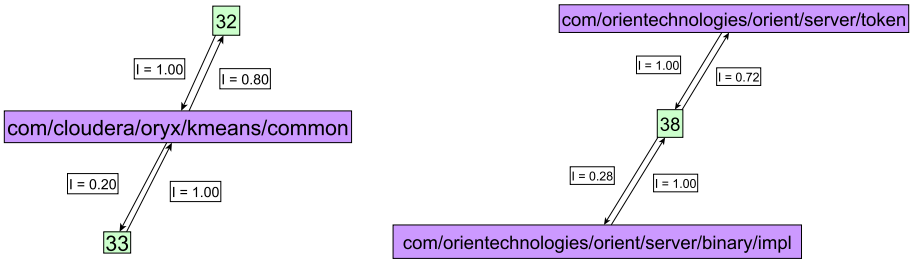



Figure 3: Examples of a clustering comparison

#### 4.2.7 com/cloudera/oryx/kmeans/computation/covariance

Another SUTS in `oryx` is the package responsible for handling covariance computations in k-means. It includes two classes called `CoMoment` and `Index` with 9 methods altogether, and the corresponding test classes with 3 test cases in total. There are 2 dynamic clusters involved, one for the first class (6 methods, 2 test cases) and one for the other class (3 methods, 1 test case).

**Possible explanation and refactoring:** In this SUTS, the static cluster was divided into two dynamic clusters. These two clusters are too small to be reasonably separated into distinct packages. Hence, the SUTS is a false positive. However, we note that in our investigation we found that this functionality might not be tested

properly, and it might have some indirect effect associated with the smell that was reported.

### 4.2.8 An example of dynamic cluster

This SUTS belongs to the subject program called `orientdb`. The corresponding part of the CSG can be seen in Figure 3(b). This pattern consists of a coverage-based cluster (with serial number 38) and two related static clusters. By investigating the content of the dynamic cluster, we found that it is mainly responsible for token handling/serialization in the OrientDB Server. It includes all 6 classes (with 61 methods) from the `server/token` package and the `OBinaryToken` class (with 27 methods) from package `server/binary/impl`. In addition, it contains 2 test classes with 9 test cases. The former package is responsible for handling and serialization of Web authentication tokens, while the latter is a bean-like class which stores information about the user, database, protocol, driver and server. This class has no direct tests, but it is covered by those test cases which examine the token serializer and token handler classes.

**Possible explanation and refactoring:** Both static or dynamic clustering results could potentially be considered for refactoring, depending on what unit test writing principles the project follows. One solution might be to use mocking to eliminate the dynamic relation between the elements of the two packages. But merging the packages (*i.e.* moving `OBinaryToken` to the `token` package) to provide a single unit is also a reasonable option in this case. Thus, the reported Structural Unit Test Smell is valid, and suggests elements that should be refactored. Furthermore, the refactoring possibilities can be directly obtained from an analysis of clusters, although choosing one requires additional information.

## 5 Threats to validity

This study contains some threats to validity. First, we selected the subjects after assuming that the integrated tests using the JUnit framework are indeed unit tests, and not other kinds of automated tests. However, during manual investigation some tests turned out to be higher level tests, and in these cases the traceability links had a slightly different meaning from that for unit tests. Also, in practice the granularity and size of a unit might differ from what is expected (a Java method). Generally speaking, it is hard to ascertain automatically whether a test is intended or not intended to be a unit test, so we verified each identified patterns manually for these properties as well. However, in actual scenarios this information will probably be known beforehand.

Another aspect to consider about the manual analysis is that this study was performed by the authors of this paper, who are experienced researchers and programmers as well. However, none of them was a developer of the given systems, hence the decisions made about the Structural Unit Test Smells and refactorings

would probably have been different if they had been made by a developer of the system.

# 6 Conclusions

In this study, we carried out an analysis of test-to-code traceability information. Unit test development has some widely accepted rules that support things like the maintenance of these tests suites. Some of them concern the structural attributes of these tests. These attributes can be described by traceability relations between the test and code. Previous studies demonstrated that fully automatic test-to-code traceability recovery is difficult, if not impossible in the general case [23, 16, 19]. There are several fundamental approaches proposed that have been proposed for this task, based on, among other things, static code analysis, call-graphs, dynamic dependency analysis, name analysis, change history and even questionnaire based approaches (see Section 2 above). However, there seems to be general agreement between researchers that no single method can provide accurate information about test and code relations.

Following this line of thinking, we developed a method that is able to detect Structural Unit Test Smells, *i.e.* locations in the code where unit test development rules are violated. In particular, we compute test-to-code traceability using two relatively straightforward automatic approaches, one based on the static physical code structure and the other on the dynamic behavior of test cases in terms of code coverage. Both can be viewed as objective descriptions of the relationship of the unit tests and code units, but from different perspectives; hence, each location where they disagree about traceability can be treated as a SUTS. Our approach is to use clustering and hence form mutually traceable groups of elements (instead of atomic traceability information), and this makes the method more robust because minor inconsistencies will probably not influence the overall results.

Here, we investigated the results of this method applied on four subject programs. Our goal was to manually check the reported Structural Unit Test Smells to see whether at least a part of these are real problems that needs to be examined. Experience indicates that most of the reported SUTSs point to parts of the test and code that could be reorganized to better follow unit test guidelines. However, in some situations it might not be worth modifying the tests and the code (*e.g.* for technical reasons). Overall, we found several typical reasons that could form the basis for future study and this might lead to an automatic classification of the Structural Unit Test Smells.

These findings have several implications. First, the method has a potential to find Structural Unit Test Smells, but the results will probably contain a large number of false positives. To filter out them, we need to carry out an investigation of the given situation. Fortunately, it seems that there are similar situations that can provide a basis for the automatic classification of the identified smells, and it may assist the developers in their refactoring activities. However, it is also clear from our manual analysis that automatic classification requires additional knowledge

(*i.e.* simply relying on the currently used static and dynamic data is not enough). Furthermore, we found several intricate SUTS patterns in the CSGs, for which we could not make informed refactoring suggestions because of their complexity and size.

Lastly, there are future possible directions for further research. One is that we could identify and automatically recognize patterns, and then propose an appropriate refactoring solution for them. Another might be the investigation of some methods that simplifies the recognition of the graph patterns, even the complex ones, where possible. The class level hierarchy and traceability relations might also be worth investigating to see whether they can provide relevant information that would help us to identify Structural Unit Test Smells.

## Acknowledgement

The project was supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

## References

[1] Atlassian. Clover Java and groovy code coverage tool homepage. `https://www.atlassian.com/software/clover/overview`. Last visited: 2017-04-06.

[2] Balogh, Gergő, Gergely, Tamás, Beszédes, Árpád, and Gyimóthy, Tibor. Are my unit tests in the right package? In *Proceedings of 16th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM'16)*, pages 137–146. IEEE, October 2016.

[3] Bavota, Gabriele, Qusef, Abdallah, Oliveto, Rocco, De Lucia, Andrea, and Binkley, David. An empirical analysis of the distribution of unit test smells and their impact on software maintenance. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, pages 56–65. IEEE, 2012.

[4] Beck, Kent, editor. *Test Driven Development: By Example*. Addison-Wesley Professional, 2002.

[5] Bertolino, Antonia. Software testing research: Achievements, challenges, dreams. In *2007 Future of Software Engineering*, pages 85–103. IEEE Computer Society, 2007.

[6] Blondel, Vincent D, Guillaume, Jean-Loup, Lambiotte, Renaud, and Lefebvre, Etienne. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P1000, 2008.

[7] Bouillon, Philipp, Krinke, Jens, Meyer, Nils, and Steimann, Friedrich. Ezunit: A framework for associating failed unit tests with potential programming

errors. *Agile Processes in Software Engineering and Extreme Programming*, pages 101–104, 2007.

[8] Breugelmans, Manuel and Van Rompaey, Bart. Testq: Exploring structural and maintenance characteristics of unit test suites. In *WASDeTT-1: 1st International Workshop on Advanced Software Development Tools and Techniques*, 2008.

[9] Bruntink, Magiel and Van Deursen, Arie. Predicting class testability using object-oriented metrics. In *Source Code Analysis and Manipulation, 2004. Fourth IEEE International Workshop on*, pages 136–145. IEEE, 2004.

[10] De Lucia, Andrea, Fasano, Fausto, and Oliveto, Rocco. Traceability management for impact analysis. In *Frontiers of Software Maintenance, 2008. FoSM 2008.*, pages 21–30. IEEE, 2008.

[11] Demeyer, Serge, Ducasse, Stéphane, and Nierstrasz, Oscar. *Object-oriented reengineering patterns*. Elsevier, 2002.

[12] Deursen, A. van, Moonen, L., Bergh, A. van den, and Kok, G. Refactoring test code. In Succi, G., Marchesi, M., Wells, D., and Williams, L., editors, *Extreme Programming Perspectives*, pages 141–152. Addison-Wesley, 2002.

[13] Feathers, Michael. *Working effectively with legacy code*. Prentice Hall Professional, 2004.

[14] Fortunato, Santo. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[15] Fowler, Martin. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[16] Gaelli, Markus, Lanza, Michele, and Nierstrasz, Oscar. Towards a taxonomy of SUnit tests. In *Proceedings of 13th International Smalltalk Conference (ISC'05)*, 2005.

[17] Hamill, Paul. *Unit Test Frameworks: Tools for High-Quality Software Development*. O'Reilly Media, Inc., 2004.

[18] Horváth, Ferenc, Vancsics, Béla, Vidács, László, Beszédes, Árpád, Tengeri, Dávid, Gergely, Tamás, and Gyimóthy, Tibor. Test suite evaluation using code coverage based metrics. In *Proceedings of the 14th Symposium on Programming Languages and Software Tools (SPLST'15)*, pages 46–60, October 2015. Also appeared in CEUR Workshop Proceedings, Vol. 1525.

[19] Kanstrén, Teemu. Towards a deeper understanding of test coverage. *Journal of Software: Evolution and Process*, 20(1):59–76, 2008.

[20] Meszaros, Gerard. *xUnit test patterns: Refactoring test code*. Pearson Education, 2007.

[21] Myers, Glenford J, Sandler, Corey, and Badgett, Tom. *The art of software testing*. John Wiley & Sons, 2011.

[22] Qusef, Abdallah, Bavota, Gabriele, Oliveto, Rocco, De Lucia, Andrea, and Binkley, Dave. Recovering test-to-code traceability using slicing and textual analysis. *Journal of Systems and Software*, 88:147–168, 2014.

[23] Rompaey, B. V. and Demeyer, S. Establishing traceability links between unit test cases and units under test. In *Software Maintenance and Reengineering, 2009. CSMR '09. 13th European Conference on*, pages 209–218, March 2009.

[24] Spanoudakis, George and Zisman, Andrea. Software traceability: a roadmap. *Handbook of Software Engineering and Knowledge Engineering*, 3:395–428, 2005.

[25] Tengeri, Dávid, Beszédes, Árpád, Havas, Dávid, and Gyimóthy, Tibor. Toolset and program repository for code coverage-based test suite analysis and manipulation. In *Proceedings of the 14th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM'14)*, pages 47–52. IEEE, September 2014.

[26] The JUnit Team. JUnit Java unit test framework homepage. `http://junit.org/`. Last visited: 2017-11-10.

[27] Van Rompaey, Bart and Demeyer, Serge. Exploring the composition of unit test suites. In *Automated Software Engineering-Workshops, 2008. ASE Workshops 2008. 23rd IEEE/ACM International Conference on*, pages 11–20. IEEE, 2008.

[28] Van Rompaey, Bart, Du Bois, Bart, and Demeyer, Serge. Characterizing the relative significance of a test smell. In *2006 22nd IEEE International Conference on Software Maintenance*, pages 391–400. IEEE, 2006.

*Received 30th May 2018*

Acta Cybernetica **23** (2018) 921–927.

# Spanning Tree Game as Prim Would Have Played*

András London[a] and András Pluhár[a]

**Abstract**

In this paper, we investigate special types of Maker-Breaker games defined on graphs. We restrict Maker's possible moves that resembles the way that was introduced by Espig, Frieze, Krivelevich and Pedgen [9]. Here, we require that the subgraph induced by Maker's edges must be connected throughout the game. Besides the normal play, we examine the biased and accelerated versions of these games.

**Keywords:** positional games, spanning tree, biased games

## 1 Introduction

In a positional game two players play on a hypergraph $\mathcal{H} = (V(\mathcal{H}), E(\mathcal{H}))$, where $E(\mathcal{H})$ is usually referred as the family of *winning sets*. The players take turns in claiming vertices of $V(\mathcal{H})$ that was not claimed previously. In the *Maker-Maker* version a player wins by claiming every elements of some edge $A \in E(\mathcal{H})$ first. In the *Maker-Breaker* version Maker wins by claiming all elements of an edge, while Breaker wins if he can prevent Maker's win. Note that a Maker-Maker game may end in a draw, while only one of the players can win a Maker-Breaker game. The players may take more than one elements in a turn; we call it an $(\mathcal{H}, a, b)$-game if the first player takes $a$ and the second takes $b$ elements. If $a = b > 1$ then it is an *accelerated* game, otherwise we call it a *biased* game. For a deeper introduction to positional games, we refer to Beck [4].

In graph games the set $V(\mathcal{H})$ is usually the edge set of a fixed graph $G$, mainly $G = K_n$, and $E(\mathcal{H})$ is a graph property. That is, Maker's goal is to build a particular structure (e.g. a spanning tree, a $K_3$ or a Hamiltonian cycle) within his own edges, while Breaker tries to prevent this. For some essential results in positional games, see e.g. [2, 3, 4]. Here we start with the classical Shannon's switching game, which is a Maker-Breaker game on the edge set of a connected graph $G$, and Maker wins

*This work was partially supported by the National Research, Development and Innovation Office - NKFIH, SNN-117879. The first author was supported by the project "Integrated program for training new generation of scientists in the fields of computer science", no EFOP-3.6.3-VEKOP-16-2017-0002. The project was supported by the European Union and co-funded by the European Social Fund.

[a]Department of Computer Science, University of Szeged
E-mail: {london,pluhar}@inf.u-szeged.hu

DOI: 10.14232/actacyb.23.3.2018.12

by taking the edges of a spanning tree. The outcome of this game is characterized by Lehman's theorem [12] stating that Maker wins (as a second player) if and only if the graph contains two edge-disjoint spanning trees.

Since the complete graph $K_n$ contains at least two disjoint spanning trees for $n \geq 4$, Shannon's switching game is trivial in this case. To make the game interesting for $K_n$, Chvátal and Erdős [7] introduced the $(1 : b)$ biased version. The outcome is a monotone function of $b$ in a sense that if Maker wins for a value $b$, and $b' < b$ then Maker also wins the $(1 : b')$-game. Similarly, if Breaker wins the $(1 : b)$-game, and $b < b'$, then Breaker wins the $(1 : b')$-game as well. Therefore they gave bounds on $b_0$, the smallest value for which Breaker wins. This turned out to be $b_0 = \Theta(n/\log n)$, which can be considered from another viewpoint. If the two players take their selection randomly in a $(a, b)$-game, then the graph consisting of Maker's edges will be similar to an element of $G(n, p)$, where $p = a/(a + b)$. However, $p = \log n/n = \Theta(1/(1 + b))$ is the threshold for connectivity, see [6]. Hence one may say the perfect and random plays result in the same outcome. This *probabilistic intuition*, or *Erdős paradigm* gives a deep insight to a game, and turned out to be true for several cases [3, 4, 5], while it is also informative when it fails [1].

Epsig *et al.* [9] brought fresh ideas to the connectivity type of games by introducing *Walker-Breaker game* and *PathWalker-Breaker game*. Walker, being located on a vertex $x$, may claim an edge $e = (x, y)$ if $e$ has not been taken by Breaker before. Upon doing this, his location is changed to $y$. PathWalker is even more restricted; he is allowed to visit a vertex only once. For Breaker's moves, there are no restrictions. Walker and PathWalker wants to visit as many vertices of $G$ as possible. It was shown that Walker (and even PathWalker) reaches at least $n - 2$ vertices of $K_n$ for large $n$. In the $1 : b$-game the number of vertices that can be visited by PathWalker falls into the interval $[n - c_1 \log n, n - c_2 \log n]$, where the values of $c_2 < c_1$ depend on only $b$.

In this study, motivated by the previous approach and some classic problems, we define new versions of Shannon's switching game. These are Maker-Breaker games where Maker's goal is to build a connected spanning subgraph of a graph $G$ such that in any moment of the game the subgraph consisting of Maker's edges is connected. We call this type of game the *PrimMaker-Breaker* game, referring to the execution of Prim's algorithm. Note that *Prim's algorithm* [16] finds a (minimal) spanning tree in a weighted undirected graph by keeping the subgraph of the already selected edges connected, in contrast to *Kruskal's algorithm* [11], which does not have this property. As a first step, we give a characterization for the $(1 : 1)$ unbiased game ( i.e. each player takes one edge per one turn). Let $H_n$ be the graph that we get from $K_{n-2,2}$ by joining the two vertices in its two-element color class, see Figure 1.

**Theorem 1.** *Playing the PrimMaker-Breaker game on a graph $G$ with $n$ vertices, PrimMaker wins as a first player if and only if $G$ contains $H_n$ as a subgraph.*

It is interesting that in both directions of the proof of Theorem 1, the actual winner may utilize a pairing strategy. Breaker's strategy can be adapted to a $(1 : b)$-game on $K_n$, and it shows that Breaker wins if $b > 1$, in contrast to the probabilistic in-

tuition which predicts $b_0 = \Omega(n/\log n)$. As it was observed before, the acceleration of games has surprising effects [14], and it may restore the probabilistic intuition destroyed by a pairing strategy in the $(1:1)$-game [1]. Here we can witness, in magnitude, a perfect restoration of that intuition.

**Theorem 2.** *Playing the $(2:b)$ PrimMaker-Breaker game on $K_n$, Maker wins if $b < n/(8\log n)$, and Breaker wins if $b > n/\ln n$.*

## 2 Background

The following result is not just one of the most important results in the theory of hypergraph games, but it can be used very effectively to decide the winner of biased hypergraph games. The case $a = b = 1$ was proved by Erdős and Selfridge in [8], and the general form was proved by Beck in [2].

**Theorem 3.** *If*

$$\sum_{A\in E(\mathcal{H})} (1+b)^{-|A|/a} < 1,$$

*then Breaker has a winning strategy in the $(\mathcal{H}, a, b)$ game.*

However, several times not Theorem 3 but its proof techniques and corollaries are used.

For the sake of a better understanding and introducing some notations, we give a sketch of the proof of case $b = 1$, and all elements of $E(\mathcal{H})$ have the same size, a more detailed proof can be found in [14].

**The uniform case with $b = 1$.** For any $A \in V(\mathcal{H})$ let $A_k(M)$ and $A_k(B)$ be the number of elements in $A$, after Maker's $k$th move, selected by Maker and Breaker, respectively. Now, for an $A \in E(\mathcal{H})$

$$w_k(A) = \begin{cases} \lambda^{A_k(M)} \text{ if } A_k(B) = 0 \\ 0 \text{ otherwise,} \end{cases}$$

where $\lambda = 2^{1/a}$. For any $x \in V(\mathcal{H})$ let $w_k(x) = \sum_{x\in A} w_k(A)$. The numbers $w_k(A)$ and $w_k(x)$ are called the *weight* of $A$ and $x$ (in the $k$th step), respectively.

In the $k$th step Breaker chooses an unselected element $y^k \in V(\mathcal{H})$ of maximum weight. Setting $w_k = \sum_{A\in E(\mathcal{H})} w_k(A)$, called the *potential*, one gets $w_k \geq w_{k+1}$, $k \geq 0$.

In particular, $w_1 \leq (\lambda^a - 1)|E(\mathcal{H})| + |E(\mathcal{H})| \leq 2|E(\mathcal{H})|$. Since $b = 1$ and the elements of $E(\mathcal{H})$ are of the same size, the inequality $\sum_{A\in E(\mathcal{H})} 2^{-|A|/a} < 1/2$ leads to the inequality $2|E(\mathcal{H})| < 2^{|A|/a}$. Let us suppose that Maker wins the game in the $k$th step. This would imply that $w_k \geq \lambda^{|A|} = 2^{|A|/a}$, contradicting the monotonicity of the potential. □

An edge $A \in E(\mathcal{H})$ is *active* if Breaker has not taken any of its elements. Conversely, $A \in E(\mathcal{H})$ is *blocked* if Breaker has already taken an element of it.

Since $w_k \leq w_1 \leq 2|E(\mathcal{H})|$ for all $k$, we have a bound on the "fill-in" of an active edge. Note that this bound holds for the non-uniform hypergraphs as well.

**Corollary 1.** *[14] Playing the Maker-Breaker $(\mathcal{H}, a, 1)$ game, Breaker may arrange that whenever $A$ is active, i.e. $A_k(B) = 0$, then $A_k(M) \leq a + a \log_2 |E(\mathcal{H})|$.*

**Proof of Corollary 1.** Just take the logarithm of the inequality $\lambda^{A_k(I)} = w_k(A) \leq w_k \leq w_1 \leq 2|E(\mathcal{H})|$ that holds for any active edge $A \in E(\mathcal{H})$. □

# 3 Proofs

**Proof of Theorem 1.** First we show if a graph $G$ on $n$ vertices contains the subgraph $H_n$, then PrimMaker wins the game as a first player. PrimMaker might restrict his moves to the edges of $H_n$ as follows. His first move is the edge $e = (u, v)$, the edge added to $K_{n-2,2}$, see Figure 1. The other edges of $H_n$ are paired such that $f, g \in E(H_n) \setminus \{e\}$ is a pair if they are incident and their common endpoint lies in $V(H_n) \setminus \{u, v\}$. PrimMaker plays according to this pairing; more precisely, in every turn, he takes one element of a pair. This keeps his subgraph connected and results in a spanning tree in the $(n-1)$st move.

In the other direction, let us assume that $G$ does not contain $H_n$, and PrimMaker's first move is an edge $e = (u, v)$. Then there must be a vertex $x \in V(G) \setminus \{u, v\}$, such that $|N(x) \cap \{u, v\}| \leq 1$. Now Breaker might also use a pairing strategy: whenever PrimMaker connects a new vertex $y$ to his subgraph, i.e. takes an edge $(z, y)$, where $z$ had been visited earlier, Breaker takes the edge $(y, x)$ if $(y, x) \in E(G)$, and moves arbitrarily otherwise. Obviously, PrimMaker can never connect the vertex $x$ to his subgraph. □

Note that we have proved a little more than was stated in Theorem 1. By winning PrimMaker builds a subgraph of diameter not more than three, which type of games was explored in [1].

**Proof of Theorem 2. PrimMaker's win**. First, we describe the winning strategy, and then show its feasibility. PrimMaker plays an equivalent auxiliary game, called the *positive minimum degree game* (see Hefetz et al. [10]), with the additional requirement that his subgraph should be connected during the game.


Figure 1: The graph $H_n$ and a possible Maker's subgraph.

PrimMaker tries to get edges incident to each vertices as quickly as possible. More precisely, he can guarantee an edge incident to the vertex $x$, before Breaker takes, say, $n/4$ edges incident to $x$. This can be achieved by an appropriate weight function method used several times before [13, 14, 15].

In order to utilize Corollary 1, we associate an auxiliary hypergraph game with the PrimMaker-Breaker game. For each vertex $x \in V(K_n)$, let the $A_x \in E(\mathcal{H})$ be the set of ordered pairs $\langle x, y\rangle$, where $y \in V(K_n) \setminus x$. That is, $A_x \cap A_y = \emptyset$ for $x \neq y$ and $|A_x| = n - 1$ for all $x \in V(K_n)$. When Maker takes the edge $(x, w)$ in the graph game, it results in taking both $\langle x, y\rangle$ and $\langle y, x\rangle$ in the hypergraph game. Of course Breaker's one move means taking $2b$ ordered pairs. Note that PrimMaker intends to play *as Breaker* in this auxiliary game.

Let us assume that PrimMaker can imitate the greedy strategy of Corollary 1 in the $(\mathcal{H}, 1, 2b)$ game. Note that in order to do so, PrimMaker *does not have to* take the pair (edge) $\langle x, y\rangle$ of the largest weight, as taking any pair from the largest weight hyperedge has the same effect on the potential function $w_k$.

Extending the notation of Corollary 1, we may say that a vertex $x$ is *blocked* if $A_x$ is blocked, i.e. PrimMaker has an edge that is incident to $x$. We shall prove by induction on the steps of the game that an arbitrary vertex can be blocked at each step. The induction hypothesis holds in the first step, and assuming it holds up to the $k$th step, we can use the bound of Corollary 1. This tells us that Breaker can take at most $b + b \log_2 n \leq n/4$ edges that are incident to an unblocked vertex $x$. Note that we can also assume that PrimMaker's edges form a tree $T_i$ after the $i$th step, and $i \leq n/2$. Indeed, in the process of blocking we never need to create cycles, so $|V(T_i)| = 2i + 1$ if the game is not already over.

Let $T_k$ be PrimMaker's graph and $U_k$ be the set of unconnected (unblocked) vertices by PrimMaker after the $k$th round, respectively. Assume that the blocking strategy requires one to block (connect) the vertex $x \in U_k$ in the $(k + 1)$th step. If there is an unoccupied edge $e = (x, y)$, $y \in T_k$, then we take it. Similarly, if there are unoccupied edges $e = (x, y)$ and $f = (y, z)$, $z \in T_k$ then we take those, and $x$ is blocked.

Assume on the contrary that there is a vertex $x \in U_k$ that cannot be blocked by PrimMaker in the $(k + 1)$st step; that is, in the subgraph of unoccupied edges there are no paths of length at most two from $x$ to $T_k$. According to the induction hypothesis, we know that Breaker has taken fewer than $n/4$ edges incident to $x$. The other endpoints of these edges cannot be in $T_k$ and actually all the edges between these endpoints and the vertices of $T_k$ are taken by Breaker. The number of these edges is at least $(n-1-n/4)(2k+1) < 3nk/2$, since $k \leq n/2$. After round $k$, Breaker has claimed $bk$ edges, therefore we should have $3nk/2 \leq bk$, which contradicts the choice of $b$.

**Breaker's win.** This direction could be deduced from the results of Chvátal and Erdős, the only difference being that they examine $(1 : b)$-game. For the sake of completeness, we sketch their proof. Breaker distributes his moves evenly. First, he puts an edge incident to all vertices, which needs no more than $n/(2b)$ rounds. During that time PrimMaker may achieve a positive degree of at most $n/b$ vertices;

these are dead for Breaker, while the others are active. Let us call the sequence of rounds a *phase* if Breaker gets a new incident edge to each (active) vertices. Breaker repeats the process above, getting a second, third, etc. edge incident to the edges that were all active at the beginning of the phase. Breaker loses the $n(1-1/b)^i \geq 1$ fraction of active vertices, so the $i$th phase is feasible if $n(1-1/b)^i \geq 1$. That is, Breaker can reach the $n$th phase if $b > n/\ln n$. But by doing so, Breaker isolates a vertex. □

# 4 Further problems

One could investigate PrimMaker-Breaker versions of any graph games, when Maker's present strategies involves disconnected edges. Sometimes the restriction of Breaker's move, introducing the notion of *PrimBreaker* seems to be a good idea.

Another possible way of defining a new game is to consider the game on a random graph, first investigated by Stojaković and Szabó [17]. For example if $V(\mathcal{H}_p) = E(G)$, where $G \in G(n,p)$ and the winning sets are the spanning trees of $G$. It was shown by Stojaković and Szabó that the probabilistic intuition applies to many of the random games, especially when there is a $p_{\mathcal{H}}$ such that if $p > p_{\mathcal{H}}$ then Maker, and if $p < p_{\mathcal{H}}$ then Breaker wins almost surely. The value of $p_{\mathcal{H}}$ should be close the threshold value of the connectivity of $G(n,p)$.

Needless to say, the PrimMaker-Breaker version of the random Shannon's switching game again defies random intuition. It would be interesting to see whether the $(2:2)$ PrimMaker-Breaker version restores it. The proof method of Theorem 2 gives only $p_{\mathcal{H}} \leq c\log n/\sqrt{n}$, although $p_{\mathcal{H}} \leq c\log n/n$ would be desirable. However, one might argue that $(2:2)$ acceleration is not enough, and $(3:3)$, or even more is needed. Another possible line is to define the PrimMaker-PrimBreaker version where the restriction of Breaker brings $p_{\mathcal{H}}$ closer to the $c\log n/n$ bound.

# References

[1] Balogh, József, Martin, Ryan, and Pluhár, András. The diameter game. *Random Structures & Algorithms*, 35(3):369–389, 2009.

[2] Beck, József. Remarks on positional games. *Acta Mathematica Hungarica*, 40(1-2):65–71, 1982.

[3] Beck, József. Deterministic graph games and a probabilistic intuition. In *Combinatorics, Geometry and Probability, a Tribute to Paul Erdős*, pages 81–94. Cambridge University Press, 1997.

[4] Beck, József. Combinatorial games: tic-tac-toe theory. In *Encyclopedia of Mathematics and its Applications*, volume 114. Cambridge University Press, 2008.

[5] Bednarska, Małgorzata and Łuczak, Tomasz. Biased positional games for which random strategies are nearly optimal. *Combinatorica*, 20(4):477–488, 2000.

[6] Bollobás, Béla. Random graphs. In *Modern Graph Theory*, pages 215–252. Springer, 1998.

[7] Chvátal, Vašek and Erdős, Paul. Biased positional games. *Annals of Discrete Mathematics*, 2:221–229, 1978.

[8] Erdős, Paul and Selfridge, John L. On a combinatorial game. *Journal of Combinatorial Theory, Series A*, 14(3):298–301, 1973.

[9] Espig, Lisa, Frieze, Alan, Krivelevich, Michael, and Pegden, Wesley. Walker-breaker games. *SIAM Journal on Discrete Mathematics*, 29(3):1476–1485, 2015.

[10] Hefetz, Dan, Krivelevich, Michael, Stojaković, Miloš, and Szabó, Tibor. Global maker–breaker games on sparse graphs. *European Journal of Combinatorics*, 32(2):162–177, 2011.

[11] Kruskal, Joseph B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[12] Lehman, Alfred. A solution of the Shannon switching game. *Journal of the Society for Industrial and Applied Mathematics*, 12(4):687–725, 1964.

[13] Pluhár, András. Generalized Harary games. *Acta Cybernetica*, 13(1):77–83, 1997.

[14] Pluhár, András. The accelerated k-in-a-row game. *Theoretical Computer Science*, 270(1-2):865–875, 2002.

[15] Pluhár, András. The recycled Kaplansky's game. *Acta Cybernetica*, 16(3):451–458, 2004.

[16] Prim, Robert Clay. Shortest connection networks and some generalizations. *Bell Labs Technical Journal*, 36(6):1389–1401, 1957.

[17] Stojaković, Miloš and Szabó, Tibor. Positional games on random graphs. *Random Structures & Algorithms*, 26(1-2):204–223, 2005.

*Received 2th May 2018*

Acta Cybernetica **23** (2018) 929–938.

# On the Advice Complexity of Coloring Bipartite Graphs and Two-Colorable Hypergraphs*

Judit Nagy-György[a]

**Abstract**

In the online coloring problem the vertices are revealed one by one to an online algorithm, which has to color them immediately as they appear. The advice complexity attempts to measure how much knowledge of the future is neccessary to achieve a given competitive ratio. Here, we examine coloring of bipartite graphs, proper and the conflict-free coloring of $k$-uniform hypergraphs and we provide lower and upper bounds for the number of bits of advice to achieve the optimal cost. For bipartite graphs the upper bound $n-2$ is tight. For the proper coloring, $n-2k$ bits are necessary and $n-2$ bits are sufficient, while for the conflict-free coloring case $n-2$ bits of advice are neccessary and sufficient to color optimally if $k > 3$.

## 1 Introduction

In this study we consider online vertex coloring. An online (hyper)graph is a structure $H^{<} = (H, <)$, where $H$ is a (hyper)graph and $<$ is a linear ordering of its vertices. We call a vertex the first, second,..., and ending vertex of an edge according to the ordering $<$. An online (hyper)graph coloring algorithm has to color the $i$-th vertex only knowing the sub(hyper)graph $H_i = (V_i, E_i)$ where $V_i$ contains the first $i$ vertices and $E_i$ contains the edges of the (hyper)graph, which are subsets of $V_i$. This means that the online algorithm receives information about the edges only when the last vertex of the edge arrives. We will use the well-known greedy algorithm **FF** (First Fit) to color the accepted vertices of the online (hyper)graphs. **FF** uses the smallest color for each vertex which does not violate the rule of the coloring. The online graph was first defined in [12], while the online hypergraph was first defined in [1].

We evaluate the efficiency of the online algorithms by the competitive ratio (see [5, 15]), where the online algorithm is compared to the optimal offline algorithm. We say that an online algorithm is $c$-competitive if its cost is at most $c$ times larger than the optimal cost.

*This study was partially supported by the ÚNKP-17-4 NEW NATIONAL EXCELLENCE PROGRAM OF THE MINISTRY OF HUMAN CAPACITIES

[a]Bolyai Institute, University of Szeged, Aradi Vértanúk tere 1, H-6720 Szeged, Hungary, E-mail: Nagy-Gyorgy@math.u-szeged.hu

DOI: 10.14232/actacyb.23.3.2018.13

Online graph coloring has been investigated in several studies, and one can find many details on the problem in the survey paper [16]. Halldórson and Szegedy in [13] showed that any online algorithm for graph coloring has a competitive ratio of $\Omega(n/\log^2 n)$. Some results are proved about algorithm **FF**. In [12] it is shown that this algorithm is the best possible on trees. In [18], an online algorithm is presented which colors $k$-colorable graphs on $n$ vertices with at most $O(n\log^{(2k-3)} n/\log^{(2k-4)} n)$ colors. The best known lower bound for the number of online colors for $k$-colorable graph on $n$ vertices $\Omega(\log^{k-1} n)$ [21]. In [17] an online algorithm is presented which colors $k$-colorable graphs on $n$ vertices with at most $O(n^{1-1/k!})$ colors.

The term advice complexity for online algorithms was introduced by the authors of [10]. The major question is the following: How many bits of advice are necessary and sufficient to obtain a competitive ratio $c$? This includes determining the number of bits to be optimal. The results of the following two sections are in the *tape model* which was introduced in [7]. In this model, the online algorithm may read an infinite advice tape written by the oracle and the advice complexity is simply the number of bits read. For more information about advice complexity, see the survey paper [6].

Mikkelsen showed in [19] that an $O(n^{1-\varepsilon})$-competitive online vertex-coloring algorithm must read $\Omega(n\log n)$ bits of advice.

Forišek et al showed in [11] that $\lceil n/2\rceil - 1$ bits of advice are needed to color optimally any online paths on $n$ vertices, and this bound is tight.

Bianchi et al proved in [4] the following theorems for bipartite graphs.

**Theorem 1** ([4])**.** *Any deterministic online algorithm needs at least $n-3$ bits of advice to color optimally every bipartite online graph on $n$ vertices.*

**Theorem 2** ([4])**.** *There exists an online algorithm which uses at most $n-2$ bits of advice to color optimally every bipartite online graph on $n$ vertices if $n > 2$.*

The advice complexity of 3-colorable graphs, 3-colorable chordal graphs and maximal outerplanar graphs were investigated in [20].

In the next section, we will improve this lower bound in Theorem 1 and provide a tight bound for the number of bits of advice for coloring bipartite graphs optimally. In section 3, we include the results on the advice complexity of proper and conflict-free coloring of two-colorable hypergraphs in the tape model. Then the last section, we will discuss the models of helper mode and answerer mode suggested by [10].

## 2 Bipartite graphs

In this section, we shall prove that $n-2$ bits of advice are neccessary to color bipartite graphs on $n$ vertices optimally. This result is tight because of Theorem 2.

The following observation will be useful.

**Observation 3.** *Suppose that $G^<$ and $H^<$ are two bipartite online graphs such that $G_i = H_i = (V_i, \emptyset)$ for some $i > 0$ and a deterministic online algorithm* **A** *colors*

*optimally both of them, and it uses advice word $w_G$ to color $G_i$ and a different advice word $w_H$ to color $H_i$. Then $w_G$ is not the prefix of $w_H$ and vice versa.*

**Theorem 4.** *Any deterministic online algorithm needs at least $n-2$ bits of advice to color every bipartite online graph on $n$ vertices optimally.*

*Proof.* For a contradiction, assume that there exists a deterministic online algorithm **A** that colors optimally every bipartite graph $G$ on at least 3 vertices using at most $|V(G)| - 3$ bits of advice. Next, consider the class of $\mathcal{G}$ of the following online graphs and set $m > 0$. For each $w \in \{0,1\}^m$, define $G_w^{\leq}$ in the following way. If $w$ consists of $m$ 1s, then the set of the vertices of $G_w = G_{\mathbf{1}}$ is $V = \{v_1, \ldots, v_{m+2}\}$ and the set of the edges is $E = \{v_i v_{m+2} \mid 1 \leq i \leq m+1\}$. Otherwise, the set of the vertices is

$$V = U_0 \cup U_1 \cup \{v_{m+2}, v_{m+3}\},$$

where

- $u_i = v_{i+1} \in U_0$ if $1 \leq i \leq m$ and the $i$th bit of $w$ is 0,
- $v_1 \in U_1$, and $u_i = v_{i+1} \in U_1$ if $1 \leq i \leq m$ and the $i$th bit of $w$ is 1.

Observe that now $U_0 \neq \emptyset$ and $U_1 \neq \emptyset$.

The set of the edges is $E = E_0 \cup E_1 \cup \{v_{m+2}v_{m+3}\}$, where $E_i = \{uv_{m+i+2} \mid u \in U_i\}$.

Next, consider the set of the advice words used for the coloring of the first $m+1$ vertices of the elements of $\mathcal{G}$ by **A** and denote it by $S$. Observe that if $s, s' \in S$ advice words used for coloring $v_1, \ldots, v_{m+1}$ then $s \neq s'$, moreover $s$ is not the prefix of $s'$, and vice versa, because these vertices form an independent set in each graph in $\mathcal{G}$ and **A** is deterministic. Also, it is easy to see that $|S|$ have to be $2^m$, and the length of each element of $S$ is at most $m$ and there is at least one $s \in S$, the advice word for $G_{\mathbf{1}}$, with length less than $m$ because of the initial assumption. But this is a contradiction by Observation 3 and the pigeonhole principle. □

# 3 2-colorable hypergraphs

A coloring of a hypergraph is an assignment of positive integers to the vertices of the hypergraph such that every edge satisfy some property. We consider two different versions of coloring. In proper hypergraph coloring each edge must contain vertices that have different colors. In conflict free (we will use the abbreviation cf) coloring each edge must contain a unique vertex which has a color different from the other vertices of the edge.

The online proper coloring of hypergraphs first was studied in [14], where it was proven that no online algorithm exists for 2-colorable $k$-uniform hypergraphs which can color them with fewer than $\lceil n/(k-1) \rceil$ colors, and it was proved that algorithm **FF** colors these hypergraphs with this many colors.

The online cf-coloring of hypergraph was defined in [8], where the authors examined the case where the input is a set of $n$ points on the line, and $R$ is the set

of the intervals of the line. They presented an algorithm which applies at most $O(\log^2(n))$ colors and they also proved a matching lower bound. The online cf-coloring of intervals was further studied in [2], where several coloring models were defined and compared. The online cf-coloring of other more general hypergraphs were studied in [3] and [9].

Now, we will present some results on $k$-uniform hypergraphs where $k > 2$ integer.

We need the definition of $\vee$-repeatable problem from [19].

**Definition 1** ([19]). *Let* P *be an online minimization problem such that for every fixed P-input, there is only a finite number of valid outputs.*

*Let $r \in \mathbb{N}$. For each $1 \leq i \leq r$, let $I_i$ be a finite set of* P*-inputs such that the followig holds: If $\sigma_1, \ldots, \sigma_r$ are such that $\sigma_i \in I_i$ for $1 \leq i \leq r$, then $\sigma = \sigma_1 \ldots \sigma_r$, where $\sigma$ obtained by concatenating the requests of the $r$ inputs, is a valid* P*-input and let $I^r = I_1 \times \ldots \times I_r = \{\sigma_1 \ldots \sigma_r | \sigma_i \in I_i, 1 \leq i \leq r\}$.*

*For each $1 \leq i \leq r$, let $\mathrm{cost}_i$ be a function which maps an output $\gamma$ for an input $\sigma \in I^r$ to a non-negative real number $\mathrm{cost}_i(\gamma, \sigma)$. We say that $\mathrm{cost}_i$ is the $i$th round cost function.*

*Let $I$ be the set of all possible request sequence for* P*. Define $\mathrm{P}^*_\vee$ to be the online problem with input $I^* = \{\sigma = (\sigma_1 \ldots \sigma_r) | r \geq 1, \sigma_i \in I\}$. An algorithm for $\mathrm{P}^*_\vee$ must produce an output $\gamma^* = (\gamma_1, \ldots, \gamma_r)$ where $\gamma_i = (y_i, \ldots, y_{n_i})$ is a valid sequence of answers for the* P*-input $\sigma_i = (x_1, \ldots, x_{n_i}) \in I$. The cost of the output $\gamma^*$ is cost $(\gamma^*, \sigma^*) = \max\{\mathrm{cost}_\mathrm{P}(\gamma_1, \sigma_1), \ldots, \mathrm{cost}_\mathrm{P}(\gamma_r, \sigma_r)\}$ where $\mathrm{cost}_\mathrm{P}(\gamma_i, \sigma_i)$ is the cost of the* P*-output $\gamma_i$ with respect to the* P*-input $\sigma_i$.*

*The optimal offline algorithm for $\mathrm{P}^*_\vee$ is denoted by $\mathrm{OPT}^*_\vee$.*

*Let $k \geq 0$. We say that* P *is strictly $\vee$-repetable with parameter $k$ if there exists a mapping $g : I^* \to I$ with the following properties:*

- *V1 For every $\sigma^* \in I^*$, $|g(\sigma^*)| \leq |\sigma^*| + k \cdot r$, where $r$ is the number of rounds in $\sigma^*$.*
- *V2 For every deterministic* P*-algorithm* ALG*, there is a deterministic $\mathrm{P}^*_\vee$-algorithm $\mathrm{ALG}^*$ such that for every $\sigma^* \in I^*$ $\mathrm{ALG}^*(\sigma^*) \leq \mathrm{ALG}(g(\sigma^*))$.*
- *V3 For every $\sigma^* \in I^*$, $\mathrm{OPT}^*(\sigma^*) \leq \mathrm{OPT}(g(\sigma^*))$.*

**Theorem 5** ([19]). *Let $P$ be a strictly $\vee$-repeatable online problem and let $I = \{\sigma_1, \ldots, \sigma_m\}$ be a finite set of* P*-inputs. Furthermore, let $t = \max_{\sigma_i \in I} \mathrm{OPT}(\sigma)$ and $\varepsilon > 0$ be a constant. Suppose that for every deterministic* P*-algorithm without advice,* ALG*, there exists some $1 \leq i \leq m$ such that $\mathrm{ALG}(\sigma_i) \geq k$. Then, for every randomized* P*-algorithm,* R*, reading $o(n)$ bits of advice, there exists a* P*-input $\sigma$ such that $\mathrm{E}(\mathrm{R}(\sigma)) \geq (1 - \varepsilon)k$ and such that $\mathrm{OPT}(\sigma) \leq t$.*

It is easy to see that our two-coloring problems are strictly $\vee$-repeatable, therefore the following corollary holds.

**Corollary 1.** *No algorithm for the online proper hypergraph coloring or the cf-coloring with $o(n)$ bits of advice can achieve a constant competitive ratio.*

### 3.1 Proper coloring

**Proposition 1.** *There exists an online algorithm which uses at most $n-2$ bits of advice to give an optimal proper coloring of every proper two-colorable $k$-uniform online hypergraph on $n$ vertices.*

*Proof.* Consider a proper two-coloring of a proper two-colorable $k$-uniform hypergraph $H^<$ on $n$ vertices. It is easy to see that the algorithm which colors the first vertex with color 1, asks for a bit of advice for each of the remaining $n-2$ vertices corresponding the parity of its color, and it colors the last vertex by **FF** colors $H^<$ optimally. □

**Theorem 6.** *Any online algorithm needs at least $n-2k$ bits of advice to give an optimal proper coloring of every proper two-colorable $k$-uniform online hypergraph on $n$ vertices if $k > 2$.*

*Proof.* Set $k > 2$ and $n \geq 2k$. Using proof by contradiction, let us assume that there exists an online algorithm **A** which uses 2 colors and fewer than $n-2k$ bits of advice for the optimal proper coloring of every proper two-colorable $k$-uniform hypergraph. Next, consider the class $\mathcal{H}_n$ of the following online hypergraphs on $n$ vertices. For each $w \in \{0,1\}^{n-2k}$, define $H_w^< \in \mathcal{H}_n$ as the set of vertices

$$V = \{v_1\} \cup U \cup X \cup Y,$$

where

- $u_i = v_{i+1} \in U$ for all $1 \leq i \leq n-2k$,
- $x_i = v_{n-2k+1+i} \in X$ for all $1 \leq i \leq k$,
- $y_i = v_{n-k+1+i} \in Y$ for all $1 \leq i \leq k-1$;

moreover, the set of the edges is

$$E = E_1 \cup E_2 \cup E_3$$

where

- $E_1 = \binom{X \cup Y \cup \{v_1\}}{k} - \{X, Y \cup \{v_1\}\}$,
- $E_2 = \{\{x_1, \ldots, x_{k-1}, u_i\} \mid \text{if the } i\text{th bit of } w \text{ is } 1\}$,
- $E_3 = \{\{y_1, \ldots, y_{k-1}, u_i\} \mid \text{if the } i\text{th bit of } w \text{ is } 0\}$.

Without loss of generality, we shall assume that both **A** and the optimal algorithm color $v_1$ with color 1. It is easy to see that there is only one proper coloring the subhypergraph induced by $X \cup Y$ by definition of $E_1$: if $x \in X$ then its color has to be 2 and the color of the elements of $Y$ have to be 1. Therefore if $v_i \in X$, then **A** has to color $v_i$ with 2 and if $v_i \in Y$, then **A** has to color $v_i$ with 1; otherwise it

cannot give a proper coloring of the subhypergraph induced by $\{v_1\} \cup X \cup Y$. Here from the definition of $E_2$ and $E_3$, the colors of vertices in $U$ are determined by $w$.

Recall that **A** uses fewer then $n-2k$ bits of advice. By definition $|\mathcal{H}_n| = 2^{n-2k}$, so by the pigeonhole principle there are at least two hypergraphs in $\mathcal{H}_n$ such that **A** cannot distinguish them when it knows only the subhypergraph induced by $V_{n-2k+1}$ because it does not contain any edge. But if $H, H' \in \mathcal{H}_n$ where $H \neq H'$, then the optimal colorings of their first $n-2k+1$ vertices are different, therefore if the advice words are the same for both of them, so **A** cannot give a proper two-coloring to both of them. □

## 3.2 Conflict-free coloring

First note that the problem of proper coloring and the problem of cf-coloring are equivalent on 3-uniform hypergraphs.

**Theorem 7.** *There exists an online algorithm which uses at most $n-2$ bits of advice to give an optimal cf-coloring of every two-cf-colorable $k$-uniform online hypergraph on $n \geq 3$ vertices if $k > 3$.*

*Proof.* Consider a two-cf-coloring of a two-cf-colorable $k$-uniform hypergraph $H^<$ on $n$ vertices. The last vertex of an edge will be called the closing vertex of the edge. Observe that if the currently appeared vertex is closing, then its color is obvious for an algorithm which knows the colors of the previous vertices. So whenever there is at most one closing vertex in the input, **FF** colors it optimally without any bit of advice. Therefore the following algorithm uses at most $n-2$ bits af advice and the coloring produced by it is optimal:

- Color the first vertex by **FF**.
- Then ask for one bit of advice. If it is 0 then use **FF** to color the remainig vertices. If it is 1 then ask for a bit of advice for every non-closing vertex and use the color whose parity is equal to this bit.
- Color any closing vertex by **FF**.

Intuitively, the $i$th advice bit indicates the color of the $(i+1)$th vertex which does not apperar as a closing vertex, if **FF** is not optimal. □

The following observation will be useful.

**Observation 8.** *Let $k > 3$ and $H$ a two-cf-colored $k$-uniform hypergraph. If we change the color of exactly one vertex in any edge, the result will not be a two-cf-coloring.*

**Corollary 2.** *Let $k > 3$ and $H$ be a two-cf-colored $k$-uniform hypergraph, $\{u_1, \ldots, u_{k-1}, u_k\}$ and $\{u_1, \ldots, u_{k-1}, u'_k\}$ be two edges of $H$ with $k-1$ common vertices. The colors of $u_k$ and $u'_k$ will be equal.*

**Theorem 9.** *Any online algorithm needs at least $n-2$ bits of advice to give an optimal cf-coloring of every two-cf-colorable $k$-uniform online hypergraph on $n$ vertices if $k > 3$.*

*Proof.* Using proof by contradiction, let us assume that there exists an algorithm **A** which uses 2 colors and fewer than $|V(H)|-2$ bits of advice for teh proper coloring of every proper two-colorable $k$-uniform hypergraph $H$. Set $m \geq 2k-2$ and consider the class of $\mathcal{H}_m$ of the following online hypergraphs. For each $w \in \{0,1\}^m$, define $H_w^<$ in the following way. If $w$ consists of $m$ 1s, then the set of the vertices of $H_w = H_{\mathbf{1}}$ is $V = \{v_1, \ldots, v_{m+2}\}$ and the set of edges is $E = \{v_{m+2} \cup U \mid U \in \binom{\{v_1,\ldots,v_{m+1}\}}{k-1}\}$. By Corollary 2 it is easy to see that the color of $v_i$ must be 1 if $1 \leq i \leq m+1$.

If $w \neq \mathbf{1}$ the set of the vertices of $H_w$ is

$$V = U_0 \cup U_1 \cup \{v_{n-1}, v_n\},$$

where

- $u_i = v_{i+1} \in U_0$ if $1 \leq i \leq m$ and the $i$th bit of $w$ is 0,
- $v_1 \in U_1$, and $u_i = v_{i+1} \in U_1$ if $1 \leq i \leq m$ and the $i$th bit of $w$ is 1.

The set of the edges is

$$E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5 \cup E_6$$

where

- $E_1 = \{\{v_{n-1}\} \cup X \mid X \in \binom{U_1}{k-1}\}$,
- $E_2 = \{\{v_{n-1}, y\} \cup Y \mid y \in U_1, X \in \binom{U_0}{k-2}\}$,
- $E_3 = \{\{v_n\} \cup X \mid X \in \binom{U_0}{k-1}\}$,
- $E_4 = \{\{v_n, y\} \cup X \mid y \in U_0, X \in \binom{U_1}{k-2}\}$,
- $E_5 = \{\{v_{n-1}, v_n\} \cup X \mid X \in \binom{U_1}{k-2}\}$,
- $E_6 = \{\{v_{n-1}, v_n\} \cup X \mid X \in \binom{U_0}{k-2}\}$,

Without loss of generality, we shall assume that both **A** and the optimal algorithm color $v_1$ with color 1. The aim is to show that there is only one two-cf-coloring of $H_w$. Note that if $n \geq 2k$, then either $E_1$ is not empty or $E_2$ and $E_3$ are not empty because $|U_0| + |U_1| \geq 2k-1$. Moreover, if $|U_0| > 0$ and $E_1 \neq \emptyset$, then $E_4 \neq \emptyset$.

For every $u \in U_1$ there are edges $e_1, e_2 \in E_1 \cup E_2$ such that the symmetric difference of these edges $e_1 \triangle e_2 = \{v_1, u\}$. Thus the color of $u$ must be 1, by Corollary 2.

For every $u, u' \in U_0$ there are edges $e_1, e_2 \in E_3 \cup E_4$ such that the $e_1 \triangle e_2 = \{u, u'\}$. Therefore the colors of $u$ and $u'$ must be the same, by Corollary 2.

Therefore the colors of $v_{n-1}$ and $v_n$ must be different, by the definition of $E_5$ and $E_6$ and Corollary 2 if $n > 2k$. Moreover, there are edges $e_1, \in E_1 \cup E_2$ and $e_2 \in E_5 \cup E_6$ such that $e_1 \triangle e_2 = \{v_n, u\}$ for any $u \in U_1$, so the color of $v_n$ must be 1 by Corollary 2 and the color of $v_{n-1}$ must be 2.

For every $u \in U_0$ there are edges $e_1, \in E_3 \cup E_4$ and $e_2 \in E_5 \cup E_6$ such that $e_1 \triangle e_2 = \{v_{n-1}, u\}$, hence the colors of $v_{n-1}$ and $u$ must be the same. We find that the colors of $u_i \in U_1 \cup U_0$ in $H_w$ are determined by $w$.

Now consider the set of the advice words used for the coloring of the first $m+1$ vertices of the elements $\mathcal{H}_m$ by $\mathbf{A}$ and denote it by $S$. Observe that if $s, s' \in S$ advice words are used for coloring $v_1, \ldots, v_{m+1}$, then $s \neq s'$; moreover, $s$ is not the prefix of $s'$ and vice versa because these vertices form an independent set in each hypergraph in $\mathcal{H}_m$ and $\mathbf{A}$ is deterministic. Also, it is easy to see that $|S|$ have to be $2^m$, the length of each element of $S$ is at most $m$ and there is at least one $s \in S$, the advice word for $H_{\mathbf{1}}$, with length less then $m$ because of the assumption. But this is a contradiction by the pigeonhole principle. □

# 4 Other models

The advice complexity was defined in [10]. The authors suggested two models, namely the *helper mode* and the *answerer mode*. These models don't use a tape. In the helper mode, the online algorithm receives a number of advice bits, which could be zero, prior to processing each request. The answerer mode is similar, except that advice bits are only given when requested by the online algorithm in which case at least one bit is given. In both of these models, fewer bits of advice are sufficient than in the tape model.

**Theorem 10.** *$o(n)$ bits of advice is sufficient to color optimally bipartite graphs / proper coloring proper two-colorable hypergraphs / cf-coloring two-cf-colorable hypergraphs on $n$ vertices in the helper and the answerer mode.*

*Proof.* Let $W_m = \{(w_1, w_2) \mid w_1 \in \{0,1\}^{m_1}, w_1 \in \{0,1\}^{m_2}, m_1 + m_2 = m\}$ and $W'_m = \{w \mid w \in \{0,1\}^{m \cdot \lfloor \log_2(m-1) \rfloor}\}$. There is an injective function $h : W'_m \to W_m$ because $|W_m| \geq m \log_2(m-1)$ and $|W'_m| = m \cdot \lfloor \log_2(m-1) \rfloor$.

Next, consider an optimal coloring of the input. Our algorithm is the following:

- First, the algorithm gets an advice word $w_1$ and colors the first vertex by **FF**.
- After the algorithm gets an advice word $w_2$ and then it colors the $(i+1)$ vertex with a color such that the parity of it is equal to the parity of the $i$th bit of $h^{-1}(w_1, w_2)$.
- The algorithm colors the remaining vertices using **FF**.

Intuitively, the $i$th bit of $h^{-1}(w_1, w_2)$ indicates the color of the $(i+1)$th vertex in the optimal coloring, if **FF** is not optimal.

Clearly, $m$ bits of advice are sufficient to color an input graph (hypergraph) on $m \cdot \lfloor \log_2 m \rfloor + 2$ vertices optimally. □

## References

[1] N. Alon, U. Arad, Y. Azar, Independent Sets in Hypergraphs with Applications to Routing Via Fixed Paths, Proc. of APPROX 99, LNCS 1671, pp 16–27, 1999

[2] A. Bar-Noy, P. Cheilaris and S. Smorodinsky, Conflict-free coloring for intervals: from offline to online, Proc. 18th Annu. ACM Sympos. Parallelism Algorithms Architectures (SPAA06), pp 128–137, 2006

[3] A. Bar-Noy, P. Cheilaris, S. Olonetsky, and S. Smorodinsky, Online conflict-free colorings for hypergraphs, Proc. of Automata, languages and programming, LNCS 4596, pp 219–230, 2007

[4] M. P. Bianchi, H-J. Böckenhauer, J. Hromkovič, L. Keller. Online Coloring of Bipartite Graphs With and Without Advice. Computing and Combinatorics, Lecture Notes in Computer Science 7434, pp 519–530, 2012

[5] A. Borodin, R. El-Yaniv, Online Computation and Competitive Analysis, Cambridge University Press, 1998

[6] J. Boyar, L. M. Favrholdt, C. Kudahl, K. S. Larsen, J. W. Mikkelsen, Online algorithms with advice: a survey, ACM Computing Surveys, 50(2), Article No. 19, 2017

[7] H-J. Böckenhauer, D. Komm, R. Královič, R. Královič, T. Mömke, On the advice complexity of online problems, ICALP (1), LNCS 6755, pp 207–218, 2011

[8] K. Chen, A. Fiat, H. Kaplan, M. Levy, J. Matousek, E. Moss el, J. Pach, M. Sharir, S. Smorodinsky, U. Wagner and E. Welzl, Online conflict-free coloring for intervals, SIAM J. Comput. 36, pp 1342–1359, 2006

[9] K. Chen, H. Kaplan, M. Sharir, Online CF coloring for halfplanes, congruent disks, and axis-parallel rectangles. ACM Transactions on Algorithms (TALG), 5(2), Article No. 16, 2009

[10] S. Dobrev, R. Královič, E. Pardubská, Measuring the problem-relevant information in input, TAIRO - Theor. Inf. Appl., 43(3), pp 585–613, 2009

[11] M. Forišek, L. Keller, M. Steinová, Advice complexity of online coloring for paths, LATA 2012, LNCS 7183, pp 228–239, 2012

[12] A. Gyárfás, J. Lehel, On-line and first-fit colorings of graphs, *Journal of Graph Theory*, **12**, pp 217–227, 1988

[13] M. M. Halldórsson, M. Szegedy, Lower bounds for on-line graph coloring, Theor. Comput. Sci., 130(1):163–174, 1994

[14] Cs. Imreh, J. Nagy-György. Online hypergraph coloring, Information Processing Letters 109(1), pp 23–26, 2008

[15] Cs. Imreh, Competitive analysis, In Algorithms of Informatics Volume 1, ed. Antal Iványi, mondAt, Budapest, pp 395–428, 2007

[16] H. A. Kierstead, Coloring Graphs On-line, in *Online algorithms: The State of the Art (A. Fiat, and G. J. Woeginger (eds.)), LNCS 1442*, pp 281–305, 1998

[17] H. A. Kierstead, On-line coloring $k$-colorable graphs, *Israel Journal of Mathematics*, **105**, pp 93–104, 1998

[18] L. Lovász, M. E. Saks, and W. T. Trotter. An on-line graph coloring algorithm sublinear performance ratio. Discrete Mathematics, 75(1–3), pp 319–325, 1989

[19] J. W. Mikkelsen, Randomization can be as helpful as a glimpse of the future in online computation, In ICALP, LIPIcs 9, pp 1–14, 2016

[20] S. Seibert, A. Sprock, W. Unger, Advice complexity of the online coloring problem, CIAC, LNCS 7878, pp 345–357, 2013

[21] S. Vishwanathan. Randomized on-line graph coloring, *Journal of Algorithms*, **13** 657–669, 1992

*Received 28th May 2018*

Acta Cybernetica **23** (2018) 939–958.

# How Sufficient Conditions are Related for Topology-Preserving Reductions

Kálmán Palágyi[a]

**Abstract**

A crucial issue in digital topology is to ensure topology preservation for reductions acting on binary pictures (i.e., operators that never change a white point to black one). Some sufficient conditions for topology-preserving reductions have been proposed for pictures on the three possible regular partitionings of the plane (i.e., the triangular, the square, and the hexagonal grids). In this paper, the relationships among these conditions are stated.

**Keywords:** digital topology, topology preservation, simple points, $P$-simple sets, hereditarily simple sets, general-simple deletion rules

## 1 Introduction

A *binary picture* on a grid is a mapping that assigns a color of *black* or *white* to each grid element called a *point* [15]. A regular partitioning of the 2D Euclidean space is formed by a tessellation of regular polygons (i.e., polygons having equal angles, and sides are all of the same length). There are exactly three polygons that can form such regular tessellations, these being the equilateral triangle, the square, and the regular hexagon [19] (see Figure 1). Although 2D digital pictures sampled on the square grid are generally assumed, triangular and hexagonal grids have also attracted significant interest [4, 15, 19, 20].

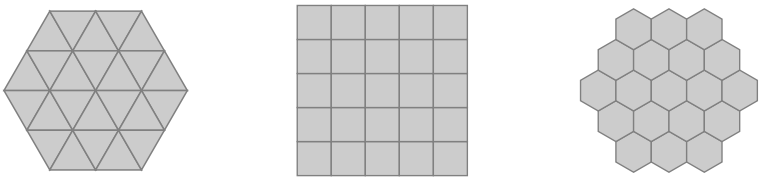


Figure 1: The three possible regular planar grids.

[a]Institute of Informatics, University of Szeged, Hungary, E-mail: palagyi@inf.u-szeged.hu

DOI: 10.14232/actacyb.23.3.2018.14

A *reduction* transforms a binary picture only by changing some black points to white ones, which is referred to as *deletion* [15]. Reductions play a key role in some *topological algorithm*s, e.g., thinning [5, 13, 15] and shrinking [6] algorithms.

*Topology preservation* is a major concern of reductions [13, 15]. In this paper, five types of sufficient conditions for topology-preserving reductions acting on the three possible regular planar grids are presented, and the relationships among these conditions are revealed.

# 2 Basic Notions and Results

In this study, we apply the fundamental concepts of digital topology as reviewed by Kong and Rosenfeld [15]. Despite the fact that there are other approaches based on cellular/cubical complexes [16], here we shall consider the 'conventional paradigm' of digital topology.

## 2.1 Binary Digital Pictures

Let us denote the triangular, the square, and the hexagonal grids by $\mathcal{T}$, $\mathbb{Z}^2$, and $\mathcal{H}$, respectively, and throughout this article, if we will use the notation $\mathcal{V}$, we will mean that $\mathcal{V}$ belongs to $\{\mathcal{T}, \mathbb{Z}^2, \mathcal{H}\}$. The elements of the given grids (i.e., regular polygons) are called *point*s. Two points are 1-*adjacent* if they share an edge and they are 2-*adjacent* if they share an edge or a vertex (see Fig. 2). Note that both relations are reflexive and symmetric. Now let us denote the set of points being $j$-adjacent to a point $p$ in the grid $\mathcal{V}$ by $N_j^{\mathcal{V}}(p)$, and let $N_j^{*\mathcal{V}}(p) = N_j^{\mathcal{V}}(p) \setminus \{p\}$ ($j = 1, 2$). It is obvious that $N_1^{\mathcal{T}}(p) \subset N_2^{\mathcal{T}}(p)$, $N_1^{\mathbb{Z}^2}(p) \subset N_2^{\mathbb{Z}^2}(p)$, and $N_1^{\mathcal{H}}(p) = N_2^{\mathcal{H}}(p)$.

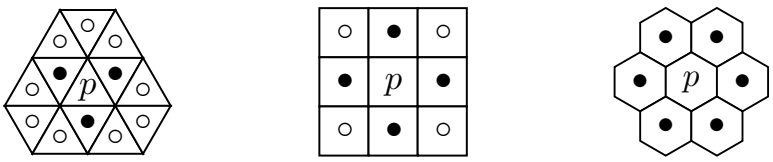



Figure 2: The adjacency relations studied on the three possible regular planar grids. Points that are 1-adjacent to the central point $p$ are marked '•', while points that are 2-adjacent but not 1-adjacent to $p$ are denoted by '○'.

A sequence of distinct points $\langle p_0, p_1, \ldots, p_m \rangle$ is called a $j$-*path* from $p_0$ to $p_m$ in a non-empty set of points $X$ if each point of the sequence is in $X$ and $p_i$ is $j$-adjacent to $p_{i-1}$ for each $i = 1, 2, \ldots, m$ ($j = 1, 2$). Two points are said to be $j$-*connected* in a set $X$ if there is a $j$-path in $X$ between them. A set of points $X$ is $j$-*connected* in the set of points $Y \supseteq X$ if any two points in $X$ are $j$-connected in $Y$. A $j$-*component* of a set of points $X$ is a maximal (with respect to inclusion) $j$-connected subset of $X$.

Let $(k, \bar{k})$ be an ordered pair of adjacency relations. Throughout this article, it is assumed that $(k, \bar{k})$ belongs to $\{(1, 2), (2, 1)\}$. A $(k, \bar{k})$ *binary digital picture* (or,

in short *picture*) is a quadruple $(\mathcal{V}, k, \bar{k}, B)$ [15], where set $\mathcal{V}$ contains all points of the given grid, $B \subseteq \mathcal{V}$ denotes the set of *black point*s, and each point in $\mathcal{V} \setminus B$ is said to be a *white point*. A *black component* or *object* is a $k$-component of $B$, while a *white component* is a $\bar{k}$-component of $\mathcal{V} \setminus B$.

Here it is assumed that a picture contains finitely many black points. Consequently there is a unique infinite white component, which is said to be the *background*. A finite white component is called a *cavity* in a picture.

A black point $p$ is an *interior point* if all points in $N_{\bar{k}}^{*\mathcal{V}}(p)$ are black. A black point $p$ is said to be a *border point* if $p$ is $\bar{k}$-adjacent to at least one white point (i.e., $N_{\bar{k}}^{*\mathcal{V}}(p) \setminus B \neq \emptyset$). A border-point $p$ is called an *isolated point* if all points in $N_k^{*\mathcal{V}}(p)$ are white (i.e., $\{p\}$ is a singleton object).

## 2.2 Topology Preservation

A reduction in a 2D picture is *topology-preserving* if each object in the input picture contains exactly one object in the output picture, and each white component in the output picture contains exactly one white component in the input picture [15]. In other words, a 2D reduction is topology-preserving if no object in the input picture is split (into two or more) or completely deleted, no cavity in the input picture is merged with the background or another cavity, and no cavity is created where there was none in the input picture [13].

Figure 3 depicts a counter-example.

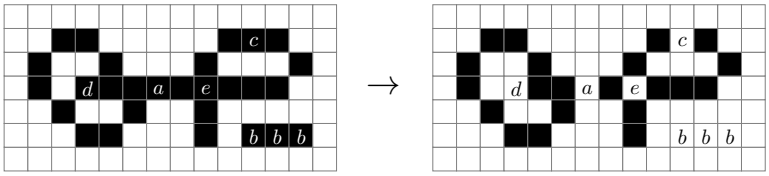



Figure 3: A reduction for a $(2, 1)$ picture on $\mathbb{Z}^2$ that is not topology-preserving. Deletion of the point marked '$a$' splits the larger object into two and the smaller object is completely deleted by deleting the points marked '$b$'; deletion of the point marked '$c$' merges a cavity with the background; the remaining two cavities are merged with each other by deleting the point '$d$'; deletion of the point marked '$e$' creates a brand new cavity.

## 2.3 Simple Points

A black point is said to be *simple* in a picture if its deletion is a topology-preserving reduction [13, 15]. In [15], Kong and Rosenfeld stated a characterization of simple points only on the square grid. Later Kardos and Palágyi stated a 'formal' and two kinds of 'easily visualized' characterizations of simple points in all the given five types of pictures on the regular 2D grids (i.e., two for $\mathcal{T}$, two for $\mathbb{Z}^2$, and one for

$\mathcal{H}$) [9, 10, 12]. The following theorem states our 'formal' necessary and sufficient condition:

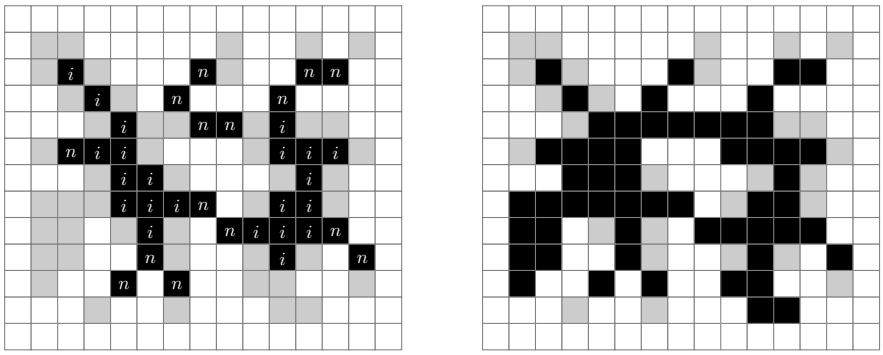
**Theorem 2.1.** [12] *Let $p$ be a black point in a picture $(\mathcal{V}, k, \bar{k}, B)$. Then $p$ is simple if and only if the following conditions hold:*

1. *$p$ is $k$-adjacent to exactly one $k$-component of $N_2^{*\mathcal{V}}(p) \cap B$.*
2. *$p$ is $\bar{k}$-adjacent to exactly one $\bar{k}$-component of $N_2^{\mathcal{V}}(p) \setminus B$.*

Theorem 2.1 shows that simplicity of a point $p$ is a local property: it can be decided by examining the set $N_2^{*\mathcal{V}}(p)$ containing just 12, 8, and 6 points for $\mathcal{T}$, $\mathbb{Z}^2$, and $\mathcal{H}$, respectively. As a straightforward consequence of the above theorem we note that if a black point is an isolated or interior point then it is not simple (i.e., some border points may be simple). Another immediate consequence of Theorem 2.1 is the following duality theorem:

**Theorem 2.2.** *A black point $p$ is simple in picture $(\mathcal{V}, k, \bar{k}, B)$ if and only if $p$ is simple in picture $(\mathcal{V}, \bar{k}, k, (\mathcal{V} \setminus B) \cup \{p\})$.*

Figure 4a classifies the set of black points in a in a $(2,1)$ picture on $\mathbb{Z}^2$ into (non-simple) interior points, non-simple border points, and simple (border) points.



(a) (b)

Figure 4: Classifying black points in a $(2,1)$ picture on $\mathbb{Z}^2$ (a). Notations: (non-simple) interior points are marked '$i$'; non-simple border points are marked '$n$'; simple (border) points are depicted in gray. An example of $P$-simple sets in the same picture (b). Elements in that $P$-simple set are depicted in gray. Note that all possible $P$-simple sets are subsets of simple points.

# 3 Sufficient Conditions for Topology-Preservation

The deletion of a single point in a picture preserves the topology if and only if it is simple in that picture. However, reductions can delete one set of black points at a

time. Hence we need a precise definition of what is meant by topology preservation when a number of points are deleted simultaneously.

**Definition 3.1.** [13, 17] *Let $B$ be the set of black points in an arbitrary picture. A set of $n$ points $Q \subset B$ is a* simple set *for $B$ if it is possible to arrange the elements of $Q$ in a sequence $\langle q_1, \ldots, q_n \rangle$ such that $q_1$ is a simple point for $B$ and each $q_i$ is simple after the set of points $\{q_1, \ldots, q_{i-1}\}$ is deleted ($i = 2, \ldots, n$). Such a sequence is called a* simple sequence. *(And let the empty set be called simple.)*

Figure 5 gives examples of simple and non-simple sets of black points in a $(2, 1)$ picture on the grid $\mathbb{Z}^2$.

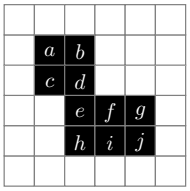



Figure 5: Examples of simple and non-simple sets in the picture $(\mathbb{Z}^2, 2, 1, \{a, \ldots, j\})$. The set of black points $\{a, b, c, d\}$ is simple as all the 12 sequences (of the possible 24 ones) $\langle a, b, c, d \rangle$, $\langle a, b, d, c \rangle$, $\langle a, c, b, d \rangle$, $\langle a, d, b, c \rangle$, $\langle b, a, c, d \rangle$, $\langle b, a, d, c \rangle$, $\langle b, c, a, d \rangle$, $\langle b, d, a, c \rangle$, $\langle c, a, b, d \rangle$, $\langle c, b, a, d \rangle$, $\langle d, a, b, c \rangle$, and $\langle d, b, a, c \rangle$ are simple. The set of black points $\{f, i\}$ is non-simple as both sequences $\langle f, i \rangle$ and $\langle i, f \rangle$ are non-simple. (Note that all black points are simple in this picture. Hence all the 10 singleton sets $\{a\}, \ldots, \{j\}$ are simple sets.)

There is general agreement that the concept of a simple set trivially implies a sufficient condition for topology-preserving reductions:

**Criterion 3.1.** [13, 17, 33] *A reduction is topology-preserving if, for all possible pictures, it deletes only simple sets.*

## 3.1 $P$-Simple Sets

Bertrand introduced the notion of a *$P$-simple set*, whose simultaneous deletion preserves the topology:

**Definition 3.2.** [1] *Let $B$ be the set of black points in an arbitrary picture. A set of black points $Q \subset B$ is a* $P$-simple set *for $B$ if for any point $q \in Q$ and any set of points $R \subseteq Q \setminus \{q\}$, $q$ is simple for $B \setminus R$. Each element of a $P$-simple set is called a* $P$-simple point.

Figure 4b shows an example of $P$-simple sets in a $(2, 1)$ picture on $\mathbb{Z}^2$.

**Theorem 3.1.** [1] *A reduction that deletes a subset composed solely of $P$-simple points is topology-preserving.*

Note that Bertrand and Couprie gave a local characterization of $P$-simple points in $(2,1)$ pictures on $\mathbb{Z}^2$ [3]. Kardos and Palágyi presented both 'formal' characterization and 'easily visualized' sufficient and necessary conditions of $P$-simple points in all the five given types of pictures [11].

## 3.2 Hereditarily Simple Sets

Kong reported an alternative solution to the problem by introducing the notion of a *hereditarily simple set*, whose simultaneous deletion is proved to be topology-preserving [13].

**Definition 3.3.** [13] *Let $B$ be the set of black points in an arbitrary picture. A set of points $Q \subset B$ is said to be* hereditarily simple *for $B$ if all subsets of $Q$ (including $Q$ itself) are simple sets in that picture.*

**Theorem 3.2.** [13] *A reduction that deletes only hereditarily simple sets is topology-preserving.*

## 3.3 Configuration-Based Condition

Ronse [33] and later Kong [13] gave a sufficient condition for topology-preserving reductions acting on $(2,1)$ pictures on $\mathbb{Z}^2$. This condition concerns some configurations of deleted points, hence it is referred to as a *configuration-based* condition. Kardos and Palágyi formulated the following unified configuration-based sufficient condition:

**Definition 3.4.** [8] *An object in a picture $(\mathcal{V}, 2, 1, B)$ is* small *if it is composed of two or more mutually* 2*-adjacent points, and it is not formed by two* 1*-adjacent points.*

**Theorem 3.3.** [8] *For any picture $(\mathcal{V}, k, \bar{k}, B)$, a reduction is topology-preserving if all of the following conditions hold.*

1. *Only simple points for $B$ are deleted.*
2. *For any two $\bar{k}$-adjacent black points $p, q \in B$ that are deleted, $p$ is simple for $B \backslash \{q\}$.*
3. *If $(k, \bar{k}) = (2,1)$, no small object is deleted completely.*

## 3.4 Point-Based Conditions

Condition 2 of Theorem 3.3 takes pairs of $\bar{k}$-adjacent deleted points into consideration, and Condition 3 applies to small objects. Hence this theorem just provides a method of verifying that a formerly constructed reduction preserves the topology, rather than a methodology for constructing topology-preserving reductions. This is why *point-based* conditions were proposed that directly provide deletion rules

of topology-preserving reductions, and allow us to construct topology-preserving thinning algorithms [21, 22].

Kardos and Palágyi proposed the following theorem that states the deletability of individual points:

**Theorem 3.4.** [9, 10, 12] *A reduction acting on $(k, \bar{k})$ pictures on $\mathcal{V}$ is topology preserving, if for any set of black points $B$ and for any point $p \in B$ that is deleted by that reduction, the following conditions hold:*

1. *Point $p$ is simple for $B$.*
2. *For any point $q \in N_{\bar{k}}^{*\mathcal{V}}(p) \cap B$ that is simple for $B$, point $p$ is simple for $B \setminus \{q\}$.*
3. *For the $(k, \bar{k}) = (2, 1)$ case, $p$ is not an element of a small object.*

Conditions of Theorem 3.4 may be viewed as *symmetric* since elements in pairs of $\bar{k}$-adjacent points (see Condition 2) and points in small objects (see Condition 3) are not distinguished.

We examined some total orderings of elements in the given three regular planar grids. Now let us assume the addressing schemes depicted in Fig. 6, which define every point in $\mathbb{Z}^2$ and $\mathcal{H}$ by a pair of coordinates and the *lexicographical order relation* '$\prec$' between two distinct points $p = (p_x, p_y)$ and $q = (q_x, q_y)$ is defined as follows: $p \prec q \Leftrightarrow (p_y < q_y) \vee ((p_y = q_y) \wedge (p_x < q_x))$. Let $Q$ be a finite set of points. Then, point $p \in Q$ is said to be the *smallest element* of $Q$ if for any $q \in Q \setminus \{p\}$, $p \prec q$.

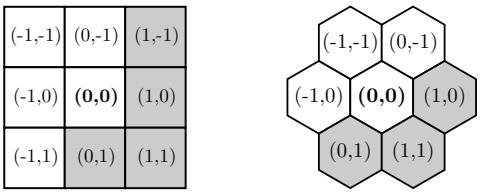



Figure 6: Feasible addressing schemes for the grids $\mathbb{Z}^2$ and $\mathcal{H}$. Each point $q$ in $N_2^{*\mathbb{Z}^2}(p)$ and $N_2^{*\mathcal{H}}(p)$ such that $p \prec q$ is depicted in gray, where $p$ is the central point with coordinates $(0, 0)$.

With the help of the proposed ordering, Kardos and Palágyi gave the following *asymmetric point-based* condition for topology-preserving reductions:

**Theorem 3.5.** [9, 10, 31] *A reduction acting on $(k, \bar{k})$ pictures on $\mathcal{V}$ is topology preserving, if for any set of black points $B$ and for any point $p \in B$ that is deleted by that reduction the following conditions hold:*

1. *Point $p$ is simple for $B$.*

2. *For any point* $q \in N_{\bar{k}}^{*\mathcal{V}}(p) \cap B$ *that is simple for* $B$ *and* $p \prec q$, *point* $p$ *is simple for* $B \setminus \{q\}$.

3. *For the* $(k, \bar{k}) = (2, 1)$ *case,* $p$ *is not the smallest element of a small object.*

Note that Kardos and Palágyi marked the smaller point in the possible pairs of $\bar{k}$-adjacent points, and the smallest point in the possible small objects on $\mathcal{T}$ [10]. Therefore relation '$\prec$' on the triangular grid has also been defined.

Our symmetric and asymmetric point-based sufficient conditions (see theorems 3.4 and 3.5) allow us to derive the following reductions:

**Definition 3.5.** *Let* $\mathsf{R}_{symm}^{\mathcal{V},(k,\bar{k})}$ *be the reduction acting on* $(k, \bar{k})$ *pictures on* $\mathcal{V}$ *that deletes all points satisfying all conditions of Theorem 3.4.*

**Definition 3.6.** *Let* $\mathsf{R}_{asymm}^{\mathcal{V},(k,\bar{k})}$ *be the reduction acting on* $(k, \bar{k})$ *pictures on* $\mathcal{V}$ *that deletes all points satisfying all conditions of Theorem 3.5.*

Note that all the five pairs of the derived reductions are evidently topology-preserving. Figure 7 gives an example of the pair of reductions acting on the hexagonal grid.

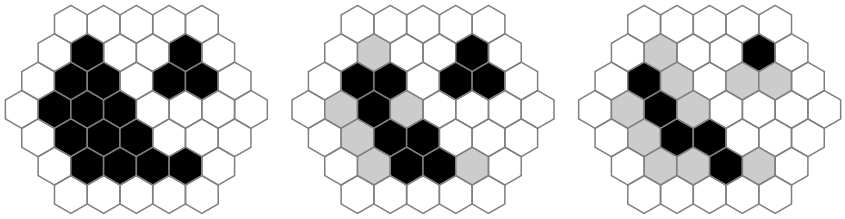


Figure 7: The original picture on the hexagonal grid $\mathcal{H}$ (left) and the results produced by the two point-based reductions $\mathsf{R}_{symm}^{\mathcal{H},(1,2)} = \mathsf{R}_{symm}^{\mathcal{H},(2,1)}$ (middle) and $\mathsf{R}_{asymm}^{\mathcal{H},(1,2)} = \mathsf{R}_{asymm}^{\mathcal{H},(2,1)}$ (right). Deleted pixels are depicted in gray.

## 3.5 General-Simple Deletion Rules

Each sufficient condition for topology-preserving reductions reported here checks some configurations of deleted points or individual deleted points. The author proposed a novel condition that considers the *deletion rule*s of reductions [23, 25, 27] that specify the points to be deleted.

*Parallel reduction*s can change a set of black points simultaneously, while *sequential reduction*s traverse the black points of a picture, and focus on the actually visited single point for possible deletion. These two absolutely dissimilar strategies are illustrated in algorithms 1 and 2.

Thinning algorithms generally classify the set of black points in input pictures into two (disjoint) subsets. That is, the deletion rule associated with a phase

**Algorithm 1:** parallel reduction

*Input*: set of black points $B$,
constraint set $C(B)$, and
deletion rule $\mathsf{R}$
*Output*: set of black points $PB$
$X = B \setminus C(B)$ *// selecting interesting points*
$D = \{\ p \mid p \in X$ and $\mathsf{R}(p, B, C(B)) = \textbf{true}\ \}$ *// determining deletable points*
$PB = B \setminus D$ *// deletion*

**Algorithm 2:** sequential reduction

*Input*: set of black points $B$,
constraint set $C(B)$,
permutation (total ordering) $\Pi$ of elements in $B \setminus C(B)$
deletion rule $\mathsf{R}$
*Output*: set of black points $SB$
$X = B \setminus C(B)$ *// selecting interesting points*
$SB = B$ *// setting initial black points*
**foreach** $p \in X$, *traversal according to* $\Pi$ **do**
  **if** $\mathcal{T}(p, SB, C(B)) = \textbf{true}$ **then**
    $SB = SB \setminus \{p\}$ *// deletion*

of an algorithm is evaluated for the elements of its set of *interesting point*s, and black points in its *constraint set* are not taken into consideration. This is why algorithms 1 and 2 treat a constraint set $C(B) \subset B$ (as an input parameter) and its complementary $X = B \setminus C(B)$ as a set of interesting points.

An interesting point $p \in X$ is *deletable* by the deletion rule $\mathsf{R}$, if $\mathsf{R}(p, Y, C(B)) = \textbf{true}$, where $Y$ denotes the set of black points in the (actual) picture, i.e., $Y = SB \subseteq B$ in sequential reductions (see Algorithm 2), and $Y = B$ in the parallel case (see Algorithm 1). Therefore, in the parallel case the initial picture is considered when the deletion rule is evaluated. In contrast, the picture is dynamically altered when a sequential reduction is performed. We should add that elements of the constraint set $C(B)$ are omitted when the deletion rule $\mathsf{R}$ is evaluated. For practical purposes, we will deal with finite pictures (i.e., $B$ contains finitely many points).

The sequential approach suffers from the drawback that different visiting orders of interesting points may yield different results. A deletion rule $\mathsf{R}$ is said to be *order-independent* if the result of Algorithm 2 is uniquely specified by $\mathsf{R}$ (i.e., the result of Algorithm 2 does not depend on the order $\Pi$ in which the points are selected by the **foreach** loop) [7, 23, 32].

Two reductions are called *equivalent* if they produce the same result for each input picture. A deletion rule is said to be *equivalent* if it yields a pair of equivalent parallel and sequential reductions.

The *support* of a deletion rule $\mathsf{R}$ applied at a point is a minimal set of points

whose values determine whether the investigated points are deleted by $\mathsf{R}$ from a picture. Note that thinning and shrinking algorithms use local supports with 'small' diameters. Let us denote the support of the deletion rule $\mathsf{R}$ with respect to a point $p$ by $\mathbf{S}_{\mathsf{R}}(p)$. (Generally $N_2^{*\mathcal{V}}(p) \subseteq \mathbf{S}_{\mathsf{R}}(p) \subseteq \bigcup_{q \in N_2^{\mathcal{V}}(p)} N_2^{\mathcal{V}}(q) \setminus \{p\}$.) It is easy to see that $\mathsf{R}(p, Y, C(B)) = \mathsf{R}(p, Y \cap \mathbf{S}_{\mathsf{R}}(p), C(B) \cap \mathbf{S}_{\mathsf{R}}(p))$.

The author introduced two special classes of deletion rules. These are:

**Definition 3.7.** [25] *Let* $\mathsf{R}$ *be a deletion rule, let* $B$ *be a set of black points in a picture, let* $p \in B \setminus C(B)$ *be an interesting point with respect to the constraint set* $C(B) \subset B$*, and let us assume that* $\mathsf{R}(p, B, C(B)) =$ **true** *(i.e.,* $p$ *can be deleted by* $\mathsf{R}$*). Then* $R$ *is* general *if* $\mathsf{R}(q, B, C(B)) = \mathsf{R}(q, B \setminus \{p\}, C(B))$ *for any point* $q \in B \setminus C(B)$*.*

In other words, a deletion rule is general if the deletability of any point does not depend on the 'color' of any deletable point. It is obvious that a method of verifying that a deletion rule $\mathsf{R}$ is general may ignore each point $q \notin \mathbf{S}_{\mathsf{R}}(p)$.

**Definition 3.8.** [25] *A deletion rule is* general-simple *if it is general, and it deletes only simple points.*

The following theorems summarize the author's most important results concerning general and general-simple deletion rules:

**Theorem 3.6.** [25] *A deletion rule* $\mathsf{R}$ *is order-independent if and only if* $\mathsf{R}$ *is general.*

**Theorem 3.7.** [25] *A deletion rule* $\mathsf{R}$ *is equivalent if* $\mathsf{R}$ *is general.*

**Theorem 3.8.** [25] *A (sequential or parallel) reduction is topology-preserving if its deletion rule is general-simple.*

Theorem 3.8 is an exceptional, sufficient condition for topology-preserving reductions. In addition, with the help of general-simple deletion rules some sequential thinning algorithms can be directly implemented for parallel computers, and conversely, some parallel algorithms can be readily implemented on conventional sequential computers.

In [24], the author proved that the deletion rule of the 2D fully parallel thinning algorithm proposed by Manzanera et al. [18] is general-simple, and Palágyi, Németh, and Kardos gave a pair of equivalent 2D sequential and parallel subiteration-based thinning algorithms [28]. Palágyi, Németh, and Kardos proposed four pairs of equivalent sequential and parallel subiteration-based 3D surface-thinning algorithms [26], and Palágyi and Németh gave a pair of equivalent sequential and fully parallel 3D surface-thinning algorithms [30].

# 4 Relationships

Next, the relationships among the given five types of sufficient conditions are presented.

## 4.1 Deletion of Hereditarily Simple Sets and Deletion of $P$-Simple Sets

In [14], Kong and Gau proved that the two kinds of sufficient conditions for topology-preserving reductions based on $P$-simple sets (i.e., Theorem 3.1) and hereditarily simple sets (i.e., Theorem 3.2) are equivalent. We will state this as a theorem:

**Theorem 4.1.** [14] *A set of black points in a picture is hereditarily simple if and only if it is a $P$-simple set in that picture.*

## 4.2 Configuration-Based and Point-Based Sufficient Conditions

Let us now state the relationship between the point-based and the configuration-based conditions:

**Theorem 4.2.** *If a reduction satisfies a point-based condition (see theorems 3.4 or 3.5), it satisfies the configuration-based condition (see Theorem 3.3) as well.*

*Proof.* It can readily be seen that if a parallel reduction satisfies Condition $i$ of Theorem 3.4 (i.e., the symmetric point-based result), Condition $i$ of Theorem 3.3 (i.e., the configuration-based result) holds for each $i \in \{1, 2, 3\}$.

Similarly, it is clear that if a parallel reduction satisfies Condition $i$ of Theorem 3.5 (i.e., the asymmetric point-based result), Condition $i$ of Theorem 3.3 (i.e., the configuration-based result) holds for each $i \in \{1, 2, 3\}$. □

## 4.3 Configuration-Based Sufficient Conditions and Deletion of $P$-Simple Sets

Palágyi and Kardos proved the following theorem:

**Theorem 4.3.** [31] *If a reduction acting on $(k, \bar{k})$ pictures on $\mathcal{V}$ deletes only $P$-simple sets, all conditions of Theorem 3.3 (i.e., the configuration-based result) are satisfied.*

We can also prove the following theorem as well:

**Theorem 4.4.** [31] *If a reduction acting on $(k, \bar{k})$ pictures on $\mathcal{V}$ satisfies all conditions of Theorem 3.3, it deletes only $P$-simple sets.*

In [31], we reported the proof of Theorem 4.4 for $(1, 2)$ pictures on $\mathbb{Z}^2$. Here, it is carried out for the hexagonal case.

By Theorem 2.1, it can readily be seen that black simple points in $(1, 2) = (2, 1)$ pictures on $\mathcal{H}$ are characterized by the matching templates depicted in Fig. 8.

Since the simplicity of a point is a local property by Theorem 2.1, the following proposition holds:


Figure 8: The five base matching templates for characterizing a black simple point $p$ in $(1,2) = (2,1)$ pictures on $\mathcal{H}$. Note that all the rotated and reflected versions of the base matching templates also match simple points.

**Proposition 4.1.** *Let $Q \subset B$ be a set of points in a picture $(\mathcal{V}, k, \bar{k}, B)$. A point $q \in Q$ is a $P$-simple point for $Q$ if for any set of points $R \subseteq N_2^{*\mathcal{V}}(q) \cap Q$, $q$ is simple for $B \setminus R$.*

**Theorem 4.5.** *If a parallel reduction obeys all the conditions of Theorem 3.3 (i.e., the configuration-based result), and it deletes the set of points $Q \subset B$ from picture $(\mathcal{H}, 1, 2, B) = (\mathcal{H}, 2, 1, B)$, then $Q$ is a $P$-simple set.*

*Proof.* Let $p \in Q$. Since Condition 1 of Theorem 3.3 holds, $p$ is simple for $B$. Without loss of generality, we will just consider the five base matching templates shown in Fig. 8.

By Proposition 4.1, the following cases are to be investigated with the help of the configurations shown in Fig. 9:

(a) If $p$ is matched by the template in Fig. 8a, then consider the configuration in Fig. 9a. In this case, only the black point $q$ need be examined. Since $p$ is a non-simple (isolated black) point in $B \setminus \{q\}$, by Condition 2 of Theorem 3.3, $q \notin Q$.

(b) If $p$ is matched by the template in Fig. 8b, then consider the configuration in Fig. 9b. Let us investigate the two black points $q$ and $r$.

- Assume that $q \in Q$ and $r \notin Q$. Since $p$ is matched by the template in Fig. 8a in $B \setminus \{q\}$, $p$ remains simple after the deletion of $q$.
- Assume that $r \in Q$ and $q \notin Q$. Since $p$ is matched by the template in Fig. 8a in $B \setminus \{r\}$, $p$ remains simple after the deletion of $r$.
- Assume that $q \in Q$ and $r \in Q$. Since $q$ is a simple point, and it remains simple after the deletion of $r$, by Condition 2 of Theorem 2.1, all points in $\{c, d, e\}$ are white. Since $r$ is a simple point, and it remains simple after the deletion of $q$, by Condition 2 of Theorem 2.1, all points in $\{a, b, c\}$ are white. Since $\{p, q, r\}$ is a small object, by Condition 3 of Theorem 3.3, we arrive at a contradiction.

(c) If $p$ is matched by the template in Fig. 8c, then consider the configuration in Fig. 9c. It can readily be seen that $p$ is not simple for $B \setminus \{r\}$. Hence, by

Condition 2 of Theorem 3.3, $r \notin Q$. Now let us examine the remaining two black points $q$ and $s$.

- Assume that $q \in Q$ and $s \notin Q$. Since $p$ is matched by the template in Fig. 8b in $B \setminus \{q\}$, $p$ remains simple after the deletion of $q$.
- Assume that $s \in Q$ and $q \notin Q$. Since $p$ is matched by the template in Fig. 8b in $B \setminus \{s\}$, $p$ remains simple after the deletion of $s$.
- Assume that $q \in Q$ and $s \in Q$. Since $p$ is matched by the template in Fig. 8a in $B \setminus \{q, s\}$, $p$ remains simple after the deletion of $\{q, s\}$.

(d) If $p$ is matched by the template in Fig. 8d, then consider the configuration in Fig. 9d. It can readily be seen that $p$ is not simple for $B \setminus \{r\}$ and $B \setminus \{s\}$. Hence, by Condition 2 of Theorem 3.3, $r \notin Q$ and $s \notin Q$. Now let us examine the remaining two black points $q$ and $t$.

- Assume that $q \in Q$ and $t \notin Q$. Since $p$ is matched by the template in Fig. 8c in $B \setminus \{q\}$, $p$ remains simple after the deletion of $q$.
- Assume that $t \in Q$ and $q \notin Q$. Since $p$ is matched by the template in Fig. 8c in $B \setminus \{t\}$, $p$ remains simple after the deletion of $t$.
- Assume that $q \in Q$ and $t \in Q$. Since $p$ is matched by the template in Fig. 8b in $B \setminus \{q, t\}$, $p$ remains simple after the deletion of $\{q, t\}$.

(e) If $p$ is matched by the template in Fig. 8e, then consider the configuration in Fig. 9e. It can readily be seen that $p$ is not simple for $B \setminus \{r\}$, $B \setminus \{s\}$, and $B \setminus \{t\}$. Hence, by Condition 2 of Theorem 3.3, $r \notin Q$, $s \notin Q$, and $t \notin Q$. Now let us examine the remaining two black points $q$ and $u$.

- Assume that $q \in Q$ and $u \notin Q$. Since $p$ is matched by the template in Fig. 8d in $B \setminus \{q\}$, $p$ remains simple after the deletion of $q$.
- Assume that $u \in Q$ and $q \notin Q$. Since $p$ is matched by the template in Fig. 8d in $B \setminus \{u\}$, $p$ remains simple after the deletion of $u$.
- Assume that $q \in Q$ and $u \in Q$. Since $p$ is matched by the template in Fig. 8c in $B \setminus \{q, u\}$, $p$ remains simple after the deletion of $\{q, u\}$.

Since $p$ remains simple after the deletion of each subset of $Q$, $p$ is a $P$-simple point for $Q$. □

In [2], Bertrand proposed a two-step (topology-preserving) thinning scheme that is based on $P$-simple points. One phase/reduction of the iterative thinning process is performed as follows:

1. A set of points $Q \subset B$ is (somehow) chosen and labeled.
2. All $P$-simple points in $Q$ are deleted (simultaneously).

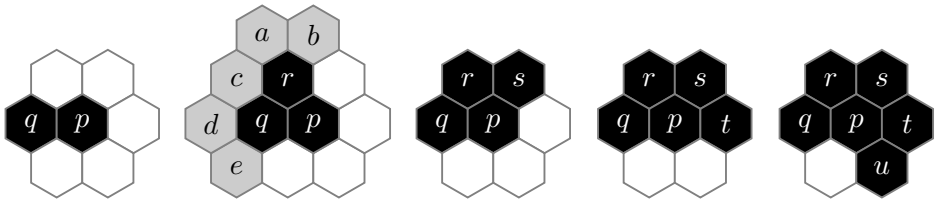



Figure 9: Configurations associated with Theorem 4.5 concerning $(1,2) = (2,1)$ pictures on $\mathcal{H}$.

Note that Step 2 concerns *tricolor* pictures (say: the value '0' corresponds to white points, the value '1' is assigned to (black) points in $B \setminus Q$, and value '2' corresponds to (black) points in $Q$). Hence this two-step scheme is both space- and time-consuming.

Theorems 4.2 and 4.4 provide a single-step thinning scheme that deletes $P$-simple points as well. The deletion rule of a reduction of the iterative thinning process can be directly constructed by combining the reduction $\mathsf{R}^{\mathcal{V},(k,\bar{k})}_{symm}$ (see Definition 3.5) or $\mathsf{R}^{\mathcal{V},(k,\bar{k})}_{asymm}$ (see Definition 3.6) with different thinning strategies (i.e., *fully parallel*, *subiteration-based*, and *subfield-based* [5]) and various geometric constraints (say *endpoints* [5]). The generated deletion rule is a common *Boolean function* that is to be evaluated for the neighborhood of the points in question in binary (*two-level*) pictures. As this Boolean function can be stored in a pre-calculated *look-up-table*, the proposed single-step scheme can be implemented efficiently.

## 4.4 Deletion of $P$-Simple Sets and General-Simple Deletion Rules

Let us consider some important properties of $P$-simple sets and general-simple deletion rules:

**Proposition 4.2.** *Let $B$ be the set of black points in an arbitrary picture, and let $p$ be an arbitrary point in a $P$-simple set $Q$ for $B$. Then $p$ is simple for $B$.*

*Proof.* Since $\emptyset \subset Q$, by Definition 3.2, $p$ is simple for $B \setminus \emptyset = B$. □

**Proposition 4.3.** *Let $B$ be the set of black points in an arbitrary picture, and let $Q$ be a $P$-simple set for $B$. Then $R$ is a $P$-simple set for $B \setminus (Q \setminus R)$ for any $R \subset Q$.*

*Proof.* Consider a point $r \in R$ and a set of points $T \subseteq R \setminus \{r\}$. Since $r \in Q$, $T \cup (Q \setminus R) \subseteq Q \setminus \{r\}$, and $Q$ is a $P$-simple set for $B$, $r$ is simple for $B \setminus (T \cup (Q \setminus R)) = (B \setminus (Q \setminus R)) \setminus T$. Hence, $R$ is a $P$-simple set for $B \setminus (Q \setminus R)$. □

**Proposition 4.4.** *Any set of points $Q \subset B$ is a $P$-simple set for $B$ if and only if all possible permutations of $Q$ form simple sequences.*

*Proof.* First, let $Q \subset B$ be a $P$-simple set of $n$ points, and consider the permutation/sequence of its elements $\langle q_1, \ldots, q_n \rangle$. Let us investigate the prefixes $\langle q_1 \rangle$, $\langle q_1, q_2 \rangle$, $\ldots$ $\langle q_1, \ldots, q_{n-1} \rangle$ of that sequence. By Proposition 4.2, point $q_1$ is simple. Since $Q$ is a $P$-simple set, and $\{q_1, \ldots, q_{m-1}\} \subseteq Q \setminus \{q_m\}$ for each $m = 2, \ldots, n$, point $q_m$ is simple for $B \setminus \{q_1, \ldots, q_{m-1}\}$. Hence, $\langle q_1, \ldots, q_n \rangle$ is a simple sequence.

Then let us assume that all possible permutations of a set $Q \subset B$ form simple sequences. Consider any point $q \in Q$ and any set of $n > 0$ points $R = \{r_1, \ldots, r_n\}$ such that $R \subseteq Q \setminus \{q\}$. Since all prefixes of a simple sequence form simple sequences, $\langle r_1, \ldots, r_n, p \rangle$ is also a simple sequence. Consequently, $q$ is a simple point for $B \setminus R$. Thus $Q$ is a $P$-simple set. □

**Proposition 4.5.** *If a deletion rule is general-simple, it is order-independent.*

*Proof.* By Definition 3.8, each general-simple deletion rule is general. Since, by Theorem 3.6, general deletion rules are order-independent, general-simple deletion rules are also order-independent. □

**Proposition 4.6.** *A deletion rule is equivalent if it is general-simple.*

*Proof.* It is actually a direct consequence of Definition 3.8 and Theorem 3.7. □

**Proposition 4.7.** *All permutations of the elements in the set of points deleted by a (sequential or parallel) reduction with a general-simple deletion rule form simple sequences.*

*Proof.* Let $\mathsf{R}$ be a general-simple deletion rule, and consider the sequential reduction (see Algorithm 2) with $\mathsf{R}$. By Theorem 3.6, Theorem 3.8, and Proposition 4.5, the sequential reduction with $\mathsf{R}$ is order-independent and topology-preserving. Consequently, the result of Algorithm 2 does not depend on the order $\Pi$ in which the points in the set of interesting points $X$ are selected in the **foreach** loop. Rule $\mathsf{R}$ is equivalent, by Proposition 4.6, hence the parallel reduction (see Algorithm 1) with $\mathsf{R}$ deletes the same set of $n$ points $D \subseteq X$. Consider the arbitrary sequence of elements in the set of $n$ points $\langle d_1, d_2, \ldots, d_n \rangle$. Since $\mathsf{R}$ deletes only simple points, $d_1$ is simple, and $d_m$ is simple after the deletion of $\{d_1, \ldots, q_{m-1}\}$ $(m = 2, \ldots, n)$. Thus $\langle d_1, d_2, \ldots, d_n \rangle$ is a simple sequence. □

We can state the following theorem as an immediate consequence of propositions 4.4 and 4.7.

**Theorem 4.6.** *Reductions with general-simple deletion rules delete $P$-simple sets.*

Now we show that the contrary statement does not hold.

**Theorem 4.7.** *The deletion rule of a reduction that deletes only $P$-simple sets may not be general-simple.*

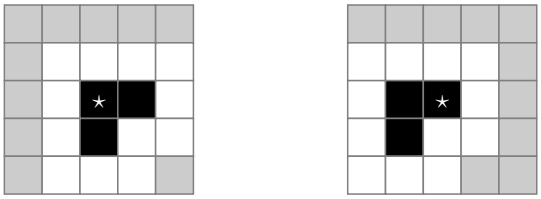


Figure 10: Matching templates associated with R working on $(2,1)$ pictures on $\mathbb{Z}^2$. The new value of a black point depends on its $5 \times 5$ neighborhood. A point is deletable by R if at least one template matches it. Notations: the position indicated by '$\star$' is the center of the template; each black element matches a black point; each white element matches a white point; each gray element matches either a black or a white point.

*Proof.* Consider the plain deletion rule R that is given by two matching templates (see Fig. 10).

It can readily be seen that the parallel reduction with R obeys all the conditions of Theorem 3.3 (i.e., the configuration-based condition for topology-preserving reductions). By Theorem 4.4, this parallel reduction deletes only $P$-simple sets.

It is obvious that the parallel reduction with R deletes both upper points of a kind of small objects (see Definition 3.4) composed of three points (and nothing else). In contrast, the sequential reduction with R can delete just one upper point that is visited first. Hence, this sequential reduction is not order-independent. Thus R is not general by Theorem 2.1, and it is not general-simple by Definition 3.8. □

Note that the author constructed a special deletion rule that deletes only $P$-simple points, and he proved that it is general-simple [29].

Lastly, we state the following theorem:

**Theorem 4.8.** *For each $P$-simple set $Q$ in a picture, there is a general-simple deletion rule that deletes $Q$ from this picture.*

*Proof.* Let $Q \subset B$ be a $P$-simple set for $B$. Consider the parallel and sequential reductions (see algorithms 1 and 2) with the following deletion rule:

$$\mathsf{R}(q, SB, B \setminus Q) = \begin{cases} \textbf{true} & \text{if } q \text{ is a } P\text{-simple point} \\ \textbf{false} & \text{otherwise} \end{cases},$$

where $SB \subseteq B$ is the set of black points in the actual picture (that is initially equal to $B$), the constraint set $C(B)$ is $B \setminus Q$, and the set of interesting points $X$ is $Q$.

It is obvious that the parallel reduction with R deletes the $P$-simple set $Q$ (and nothing else).

To prove this theorem, it is necessary to show that R is general-simple. By Proposition 4.2, R deletes only simple points. Hence the only thing we need to verify is that deletion rule R is general.

Consider a set of points $D \subseteq Q$, and two points $p, q \in Q \setminus D$, and let us assume that $SB = B \setminus D$. Since $Q \subset B$ is a $P$-simple set for $B$, by Proposition 4.3, both points $p$ and $q$ are $P$-simple for $SB$, and $q$ are $P$-simple for $SB \setminus \{p\}$. Consequently, $\mathsf{R}(p, SB, B \setminus Q) = \mathbf{true}$ and $\mathsf{R}(q, SB, B \setminus Q) = \mathsf{R}(q, SB \setminus \{p\}, B \setminus Q)$. Thus $\mathsf{R}$ is general. □

### 4.5 Summary of Relationships

Here, we summarize the relationships among the five types of sufficient conditions for topology-preserving reductions with the help of Fig. 11. Note that three of them (namely: deletion of $P$-simple sets, deletion of hereditarily simple sets, and general-simple deletion rules) are absolutely universal, and the relationships among them are valid for arbitrary pictures.

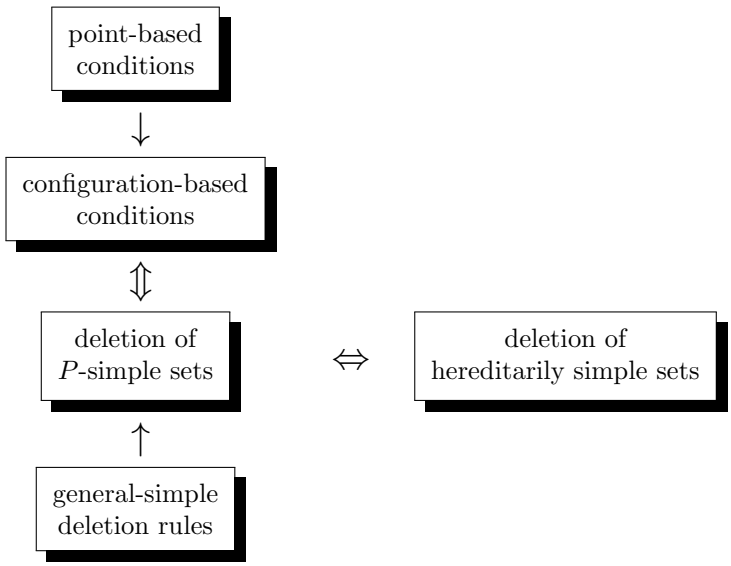



Figure 11: How the five kinds of sufficient conditions for topology-preserving reductions are related.

The linkage between $P$-simple sets and hereditarily simple sets was established by Kong and Gau [14], and the remaining relationships were discovered by Palágyi and Kardos.

## 5 Conclusions

In this paper, five types of sufficient conditions for topology-preserving reductions acting on the three possible regular planar grids are reported, and relationships among these conditions were presented. These conditions are based on configurations, individual deletable points, $P$-simple sets, hereditarily simple sets, and

general-simple deletion rules. The given sufficient conditions are absolutely not autotelic, they provide methods of verifying that a reduction preserves the topology, allow us to generate topology-preserving reductions, and they provide computationally efficient thinning algorithms.

# References

[1] Bertrand, G. On $P$-simple points. *Compte Rendu de l'Académie des Sciences de Paris, Série Math.*, 321:1077–1084, 1995.

[2] Bertrand, G. $P$-simple points: A solution for parallel thinning. In: *Proc. 5th Int. Conference on Discrete Geometry for Computer Imagery, DGCI 2005*, pages 233–242, 1995.

[3] Bertrand, G., Couprie, M. Two-dimensional parallel thinning algorithms based on critical kernels. *Journal of Mathematical Imaging and Vision*, 31:35–56, 2008.

[4] Brimkov, V.E., Barneva, R.P. Analytical honeycomb geometry for raster and volume graphics. *The Computer Journal*, 48:180–199, 2005.

[5] Hall, R.W. Parallel connectivity-preserving thinning algorithms. In Kong, T.Y., Rosenfeld, A., editors, *Topological algorithms for digital image processing*, pages 145–179, Amsterdam, 1996. Elsevier Science B.V.

[6] Hall, R.W., Kong, T.Y., Rosenfeld, A. Shrinking binary images. In Kong, T.Y., Rosenfeld, A., editors, *Topological algorithms for digital image processing*, pages 31–98, Amsterdam, 1996. Elsevier Science B.V.

[7] Kardos, P., Palágyi, K. Order-independent sequential thinning in arbitrary dimensions. In: *Proc. Int. Conference on Signal and Image Processing and Applications, SIPA 2011*, pages 129–134, 2011.

[8] Kardos, P., Palágyi, K. On topology preservation in triangular, square, and hexagonal grids. In: *Proc. 8th Int. Symposium on Image and Signal Processing and Analysis, IEEE/EURASIP, ISPA 2013*, pages 782–787, 2013.

[9] Kardos, P., Palágyi, K. Topology-preserving hexagonal thinning. *Int. Journal of Computer Mathematics*, 90:1607–1617, 2013.

[10] Kardos, P., Palágyi, K. Topology preservation on the triangular grid. *Annals of Mathematics and Artificial Intelligence*, 75:53–68, 2015.

[11] Kardos, P., Palágyi, K. Unified characterization of $P$-simple points in triangular, square, and hexagonal grids. In: *Proc. Int. Symposium on Computational Modeling of Objects Presented in Images: Fundamentals, Methods, and Applications, CompIMAGE'16, LNCS 10149*, pages 79–88, 2016, Springer.

[12] Kardos, P., Palágyi, K. On topology preservation of mixed operators in triangular, square, and hexagonal grids. *Discrete Applied Mathematics*, 216: 441–448, 2017.

[13] Kong, T.Y. On topology preservation in 2-D and 3-D thinning. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 9:813–844, 1995.

[14] Kong, T.Y., Gau, C-J. Minimal non-simple sets in 4-dimensional binary images with (8,80)-adjacency. In: *Combinatorial Image Analysis, IWCIA 2004, LNCS 3322*, pages 318–333, 2004, Springer.

[15] Kong, T.Y., Rosenfeld, A. Digital topology: Introduction and survey. *Computer Vision, Graphics, and Image Processing*, 48:357–393, 1989.

[16] Kovalevsky, V.A. Geometry of locally finite spaces. Publishing House, 2008.

[17] Ma, C.M. On topology preservation in 3D thinning. *CVGIP: Image Understanding*, 59:328–339, 1994.

[18] Manzanera, A., Bernard, T.M., Pretêux, F., Longuet, B. $n$-dimensional skeletonization: a unified mathematical framework. *Journal of Electronic Imaging*, 11:25–37, 2002.

[19] Marchand-Maillet, S., Sharaiha, Y.M. Binary digital image processing – A discrete approach. Academic Press, 2000.

[20] Middleton, L., Sivaswamy, J. Hexagonal image processing: A practical approach, Advances in Pattern Recognition. Springer-Verlag, 2005.

[21] Németh, G., Kardos, P., Palágyi, K. 2D parallel thinning and shrinking based on sufficient conditions for topology preservation. *Acta Cybernetica*, 20:125–144, 2011.

[22] Németh, G., Palágyi, K. Topology preserving parallel thinning algorithms. *International Journal of Imaging Systems and Technology* 21:37–44, 2011.

[23] Palágyi, K. Deletion rules for equivalent sequential and parallel reductions. In: *Proc. 18th Iberoamerican Congress on Pattern Recognition, Part I, CIARP 2013, LNCS 8285*, pages 17–24, 2013, Springer.

[24] Palágyi, K. Equivalent 2D sequential and parallel thinning algorithms. In: *Proc. 16th Int. Workshop on Combinatorial Image Analysis, IWCIA 2014, LNCS 8466*, pages 91–100, 2014, Springer.

[25] Palágyi, K. Equivalent sequential and parallel reductions in arbitrary binary pictures. *Int. J. Pattern Recognition and Artificial Intelligence*, 28:1460009-1–1460009-16, 2014.

[26] Palágyi, K., Németh, G., Kardos, P. Equivalent sequential and parallel subiteration-based surface-thinning algorithms. In: *Proc. 17th Int. Workshop on Combinatorial Image Analysis, IWCIA 2015, LNCS 9448*, pages 31–45, 2015, Springer.

[27] Palágyi, K. Topology-preserving general operators in arbitrary binary pictures. In: *Proc. 19th Iberoamerican Congress on Pattern Recognition, CIARP 2014, LNCS 8827*, pages 22–29, 2014, Springer.

[28] Palágyi, K., Németh, G., Kardos, P. Topology-preserving equivalent parallel and sequential 4-subiteration 2D thinning algorithms. In: *Proc. 9th Int. Symposium on Image and Signal Processing and Analysis, IEEE/EURASIP, ISPA 2015*, pages 306–311, 2015.

[29] Palágyi, K. $P$-simple points and general-simple deletion rules. In: *Proc. 19th IAPR Int. Conference on Discrete Geometry for Computer Imagery, DGCI 2016, LNCS 9647*, pages 143–153, 2016, Springer.

[30] Palágyi, K., Németh, G. A pair of equivalent sequential and fully parallel 3D surface-thinning algorithms. *Discrete Applied Mathematics*, 216:348–361, 2017.

[31] Palágyi, K., Kardos, P. A single-step 2D thinning scheme with deletion of $P$-simple points. In: *Proc. 22nd Iberoamerican Congress on Pattern Recognition, CIARP 2017, LNCS 10657*, in press.

[32] Ranwez, V., Soille, P. Order independent homotopic thinning for binary and grey tone anchored skeletons. *Pattern Recognition Letters*, 23:687–702, 2002.

[33] Ronse, C. Minimal test patterns for connectivity preservation in parallel thinning algorithms for binary digital images. *Discrete Applied Mathematics*, 21:67–79, 1988.

*Received 2nd May 2018*

Acta Cybernetica **23** (2018) 959–979.

# On Eigenvectors of the Pascal and Reed-Muller-Fourier Transforms*

Tamás Waldhauser[a]

**Abstract**

In their paper at the International Symposium on Multiple-Valued Logic in 2017, C. Moraga, R. S. Stanković, M. Stanković and S. Stojković presented a conjecture for the number of fixed points (i.e., eigenvectors with eigenvalue 1) of the Reed-Muller-Fourier transform of functions of several variables in multiple-valued logic. We will prove this conjecture, and we will generalize it in two directions: we will deal with other transforms as well (such as the discrete Pascal transform and more general triangular self-inverse transforms), and we will also consider eigenvectors corresponding to other eigenvalues.

**Keywords:** Reed-Muller-Fourier transform, discrete Pascal transform, eigenvector, eigenvalue, fixed point, multiple-valued logic, functions of several variables

## 1 Introduction

In multiple-valued logic, one of the main objects of study is functions of several variables defined on a finite set of logical values. If the number of values is $h$, then it is natural to represent them as elements of $\mathbb{Z}_h$, the ring of residue classes of integers modulo $h$, so that arithmetical operations can be performed. The case $h = 2$ corresponds to Boolean functions, which can be represented by polynomials over the two-element field $\mathbb{Z}_2$. This Reed-Muller representation [9, 11] of Boolean functions (also discovered earlier by Zhegalkin [19, 20]) has several generalizations to the multiple-valued case, one of them being the Reed-Muller-Fourier transform [13], which is also an extension of the instant Fourier transform of Gibbs [3]. We give the definition of the Reed-Muller-Fourier transform in Section 2; and for more information, we refer the reader to [14, 15, 16].

Aburdene and Goodman defined a seemingly unrelated transform, the so-called discrete Pascal transform [1], which has applications in image and signal processing

*This study was supported by the Hungarian National Research, Development and Innovation Office (NKFIH grant no. K115518).

[a]Bolyai Institute, University of Szeged, Aradi vértanúk tere 1, H–6720 Szeged, Hungary, E-mail: twaldha@math.u-szeged.hu

DOI: 10.14232/actacyb.23.3.2018.15

[1, 4, 17]. It was noticed in [6] that the above two transforms are strongly related: the Reed-Muller-Fourier transform of one-variable functions is essentially the same as the Pascal transform (see Section 2 for details).

A common feature of the two transforms is that they can be given by lower triangular self-inverse matrices over $\mathbb{Z}_h$, i.e., they are of the form $\mathbf{v} \mapsto S\mathbf{v}$, where $\mathbf{v} \in \mathbb{Z}_h^N$, and $S \in \mathbb{Z}_h^{N\times N}$ is a lower triangular matrix such that $S^2 = I_N$. This implies that if $\mathbf{v}$ is an eigenvector corresponding to the eigenvalue $\lambda$, then $\mathbf{v} = S^2\mathbf{v} = \lambda^2\mathbf{v}$. Therefore, it is natural to consider eigenvalues $\lambda$ such that $\lambda^2 = 1$, although other eigenvalues might also exist (see Example 2.1 and Table 8). The self-inverse property means that the (permutation of $\mathbb{Z}_h^N$ induced by the) transform consists of cycles of length 2 and 1; therefore, the number of fixed points completely determines the cycle structure.

The eigenfunctions of the Reed-Muller transform of Boolean and multiple-valued functions were examined in [12] and [8], respectively. For the Reed-Muller-Fourier transform, the study of the eigenfunctions was initiated in [7], and the following conjecture was formulated about the number of fixed points (note that it agrees with the result of [12] for $h = 2$).

**Conjecture 1.1** ([7])**.** *For all natural numbers $h \geq 2$ and $n \geq 1$, the number of fixed points of the Reed-Muller-Fourier transform of $n$-variable functions defined on an $h$-element domain is $h^{\lfloor h^n/2 \rfloor}$ if $n$ is odd, and it is $h^{\lceil h^n/2 \rceil}$ if $n$ is even.*

The main goal of this study is to prove the above conjecture, and, more generally, determine the number of eigenvectors corresponding to eigenvalues $\lambda$ with $\lambda^2 = 1$. After presenting the required definitions and tools in Section 2, we will prove in Section 3 that if $h$ is odd and $\lambda \in \mathbb{Z}_h$ satisfies $\lambda^2 = 1$, then the number of eigenvectors corresponding to the eigenvalue $\lambda$ of an arbitrary triangular self-inverse matrix $S \in \mathbb{Z}_h^{N\times N}$ depends only on the diagonal entries of $S$ (Theorem 3.1). This result already proves Conjecture 1.1 for odd $h$. Let us add that this case was also settled in [18] using a different method. The results of [18] also indicate that the space of fixed points has a basis, which is not true for arbitrary subspaces of $\mathbb{Z}_h^N$ (see Example 2.1). The proof presented here does not provide the existence of a basis, but it is simpler and more general than the proof in [18].

One can easily find examples showing that if $h$ is even, then it is not sufficient to know the diagonal entries of $S$ in order to determine the number of eigenvectors. Therefore, in sections 4 and 5 we deal with the Pascal transform and the Reed-Muller-Fourier transform separately. The main results are Theorem 4.1 and Theorem 5.1, which give the number of eigenvectors of these transforms corresponding to eigenvalues $\lambda$ such that $\lambda^2 = 1$. As a corollary, we get the number of fixed points of the Reed-Muller-Fourier transform (Corollary 5.1), which in turn proves Conjecture 1.1.

## 2 Preliminaries

We will work with vectors and matrices over $\mathbb{Z}_h$, the ring of integers modulo $h$ (with $h \geq 2$); thus, our methods will be of a linear algebraic flavor. However, if $h$ is a composite number, then $\mathbb{Z}_h$ is not a field, and $\mathbb{Z}_h^N$ is not a vector space, but just a module, and some familiar facts from linear algebra do not hold in this case. Nevertheless, we will use the more familiar linear algebraic terminology; for instance, we will talk about subspaces instead of submodules. By a *subspace* of $\mathbb{Z}_h^N$ we mean a set $U \subseteq \mathbb{Z}_h^N$ that is closed under *linear combinations*, i.e., $\alpha_1\mathbf{u}_1 + \cdots + \alpha_k\mathbf{u}_k \in U$ for all $\mathbf{u}_1, \ldots, \mathbf{u}_k \in U$ and $\alpha_1, \ldots, \alpha_k \in \mathbb{Z}_h$. Example 2.1 demonstrates that there exist subspaces that do not have a basis. If a subspace $U$ does have a basis of cardinality $d$, then $|U| = h^d$, since every element of $U$ can be expressed uniquely as a linear combination of the basis vectors. This shows that the size of the basis (if it exists) is uniquely determined.

We shall not make any sharp distinction between an integer $a \in \mathbb{Z}$ and the modulo $h$ residue class $a \in \mathbb{Z}_h$ containing $a$; we will use the same notation for them, but the context should make it clear which one is meant. If, occasionally, we need to use residues with respect to a modulus different from $h$, then we will write congruence instead of equality, indicating the modulus explicitly. We will use the following elementary fact without further mention: A linear equation $ax = b$ has a solution $x \in \mathbb{Z}_h$ if and only if $\gcd(a, h)$ divides $b$, and then the number of solutions is $\gcd(a, h)$. In particular, an element $a \in \mathbb{Z}_h$ has a multiplicative inverse if and only if $a$ and $h$ are relatively prime, and the inverse is unique. Consequently, if the determinant of a matrix $S \in \mathbb{Z}_h^{N\times N}$ is relatively prime to $h$, then $S$ has an inverse matrix $S^{-1} \in \mathbb{Z}_h^{N\times N}$. In particular, if $S$ is a (lower or upper) triangular matrix such that each entry on its diagonal is $\pm 1$, then $S$ has an inverse.

We say that a nonzero vector $\mathbf{u} \in \mathbb{Z}_h^N$ is an *eigenvector* of $S \in \mathbb{Z}_h^{N\times N}$ corresponding to the *eigenvalue* $\lambda \in \mathbb{Z}_h$, if $S\mathbf{u} = \lambda\mathbf{u}$. (Here, and in the sequel, all vectors will be considered as column vectors.) The set of all eigenvectors corresponding to $\lambda$ together with the zero vector $\mathbf{0}$ form the *eigenspace* $U_\lambda(S) = \{\mathbf{u} \in \mathbb{Z}_h^N : S\mathbf{u} = \lambda\mathbf{u}\} \leq \mathbb{Z}_h^N$. (We will often omit the matrix $S$ from the notation, when there is no risk of ambiguity.)

Let $P_N$ be the matrix obtained by arranging the first $N$ rows of the Pascal triangle in a lower triangular matrix with every second column multiplied by $-1$ (see Table 1). Formally,

$$P_N = (p_{ij})_{i,j=0}^{N-1} \in \mathbb{Z}_h^{N\times N}, \text{ where } p_{ij} = (-1)^j \cdot \binom{i}{j}.$$

Note that we start the numbering of rows and columns by zero; in particular, we refer to the top row of a matrix as "row 0 ". The *discrete Pascal transform* is simply the linear transformation $\mathbb{Z}_h^N \to \mathbb{Z}_h^N$, $\mathbf{u} \mapsto P_N\mathbf{u}$ induced by the matrix $P_N$. It is not hard to see that $P_N$ is a *self-inverse* matrix, i.e., $S_N^2 = I_N$, where $I_N$ denotes the $N \times N$ identity matrix.

For the definition of the Reed-Muller-Fourier transform, we need the notion of the Kronecker product of matrices. If $A = (a_{ij}) \in \mathbb{Z}_h^{m\times n}$ and $B = (b_{ij}) \in \mathbb{Z}_h^{r\times s}$

are matrices of arbitrary sizes, then their *Kronecker product* is the $mr \times ns$ block matrix

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}.$$

The Kronecker product is associative but not commutative, it is distributive over sums, and it satisfies the following mixed product identity (for arbitrary matrices $A, B, C, D$ of appropriate sizes so that both sides are defined):

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD). \tag{1}$$

We will need the following technical lemma about eigenspaces of certain Kronecker products.

**Lemma 2.1.** *Let $p$ be a prime number, and let $A \in \mathbb{Z}_p^{n\times n}$ be a lower triangular matrix such that every diagonal entry of $A$ is 1. Then for every square matrix $B \in \mathbb{Z}_p^{m\times m}$ and $\lambda \in \mathbb{Z}_p$, we have the following inequality between the dimensions of the eigenspaces of $B$ and of $A \otimes B$:*

$$\dim U_\lambda(A \otimes B) \le n \cdot \dim U_\lambda(B).$$

*Proof.* We are working over $\mathbb{Z}_p$, which is a field, so we can use standard linear algebra; in particular, we can speak of the dimension of a subspace, as every subspace has a basis. Let us denote the rank of the matrix $B - \lambda I_m$ by $r$. Note that the eigenspace $U_\lambda(B)$ is the kernel (nullspace) of $B - \lambda I_m$, and its dimension is called the nullity of $B - \lambda I_m$. The so-called rank-nullity theorem asserts that the sum of the rank and the nullity of $B - \lambda I_m$ equals $m$, thus $\dim U_\lambda(B) = \dim \ker(B - \lambda I_m) = m - r$.

Since $\operatorname{rank}(B - \lambda I_m) = r$, one can choose rows $i_1, \dots, i_r$ and columns $j_1, \dots, j_r$ of $B - \lambda I_m$ such that the $r \times r$ submatrix $S$ of $B - \lambda I_m$ that is formed by the intersections of these rows and columns has a nonzero determinant. Let us choose the corresponding rows of $A \otimes B - \lambda I_{nm}$ in each "copy" of $B$:

$$i_1, \dots, i_r,\ i_1 + m, \dots, i_r + m, \dots, i_1 + (n-1)m, \dots, i_r + (n-1)m.$$

Similarly, let us choose the following columns:

$$j_1, \dots, j_r,\ j_1 + m, \dots, j_r + m, \dots, j_1 + (n-1)m, \dots, j_r + (n-1)m.$$

The intersections of these rows and columns of $A \otimes B - \lambda I_{nm}$ (see the gray squares in Figure 1) form an $nr \times nr$ submatrix $\tilde{S}$ that has the following structure (each $0_r$ denotes an $r \times r$ zero matrix):

$$\tilde{S} = \begin{pmatrix} S & 0_r & \cdots & 0_r \\ * & S & \cdots & 0_r \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \cdots & S \end{pmatrix}. \tag{2}$$

The assumption that each entry on the diagonal of $A$ is 1 implies that $A \otimes B$ has $n$ copies of $B$ on its diagonal, hence $A \otimes B - \lambda I_{nm}$ has $n$ copies of $B - \lambda I_m$ on its diagonal. Therefore, $\tilde{S}$ indeed has $n$ copies of $S$ on its diagonal, as shown in (2).

We see that the matrix $A \otimes B - \lambda I_{nm}$ has the $nr \times nr$ submatrix $\tilde{S}$ with $\det(\tilde{S}) = \det(S)^n \neq 0$, hence $\operatorname{rank}(A \otimes B - \lambda I_{nm}) \geq nr$. Using the rank-nullity theorem for $A \otimes B - \lambda I_{nm}$, we see that

$$\begin{aligned}\dim U_\lambda(A \otimes B) &= \dim \ker(A \otimes B - \lambda I_{nm}) \\ &= nm - \operatorname{rank}(A \otimes B - \lambda I_{nm}) \\ &\leq nm - nr = n(m-r) = n \cdot \dim U_\lambda(B).\end{aligned}$$

□

Let $T_h = -P_h$ (see Table 2), and let $T_h^{\otimes n} \in \mathbb{Z}_h^{h^n \times h^n}$ be the $n$-fold Kronecker product of $T_h$ with itself: $T_h^{\otimes n} = T_h \otimes \cdots \otimes T_h$ (see tables 3, 4 and 5 for some examples). The entries of $T_h$ are

$$t_{ij} = -p_{ij} = (-1)^{j+1} \cdot \binom{i}{j};$$

for an explicit formula for the entries of $T_h^{\otimes n}$, see the proof of Proposition 2.1 below. The mixed product identity (1) shows that $T_h^{\otimes n}$ is also a self-inverse matrix.

Listing all values of an $n$-variable function $f\colon \mathbb{Z}_h^n \to \mathbb{Z}_h$, we obtain a vector of length $h^n$, which uniquely determines $f$. More precisely, let us define the *value vector* of $f$ as the column vector $\mathbf{v}_f \in \mathbb{Z}_h^{h^n}$ consisting of the values $f(\mathbf{x})$ listed in the lexicographic order of $\mathbf{x} \in \mathbb{Z}_h^n$:

$$\mathbf{v}_f = (f(0,0,\ldots,0), f(0,0,\ldots,1), \ldots, f(h-1,h-1,\ldots,h-1))^T.$$

The *Reed-Muller-Fourier transform* of $f$ is then defined as the unique function $\mathrm{RMF}(f) : \mathbb{Z}_h^n \to \mathbb{Z}_h$ whose value vector is $T_h^{\otimes n}\mathbf{v}_f$:

$$\mathbf{v}_{\mathrm{RMF}(f)} = T_h^{\otimes n}\mathbf{v}_f.$$

Lucas' theorem about binomial coefficients modulo a prime implies that if $h$ is a prime number, then the relationship between the Reed-Muller-Fourier transform and the Pascal transform stated in [6] for $n = 1$ holds in fact for every $n$.

**Proposition 2.1.** *If $h$ is a prime number, then $T_h^{\otimes n} = (-1)^n \cdot P_{h^n}$ for all natural numbers $n$.*

*Proof.* Let us consider the representation of $i, j \in \{0, 1, \ldots, h^n - 1\}$ in the $h$-ary number system: $i = i_0 + i_1 h + \cdots + i_{n-1}h^{n-1}$ and $j = j_0 + j_1 h + \cdots + j_{n-1}h^{n-1}$, where $i_k, j_k \in \{0, 1, \ldots, h-1\}$ for $k = 0, 1, \ldots, n-1$. It follows from the definition of the Kronecker product that $\left(T_h^{\otimes n}\right)_{ij} = t_{i_0 j_0} \cdot t_{i_1 j_1} \cdot \ldots \cdot t_{i_{n-1} j_{n-1}}$. Therefore,

$$\begin{aligned}\left(T_h^{\otimes n}\right)_{ij} &= (-1)^{j_0+1} \cdot \binom{i_0}{j_0} \cdot (-1)^{j_1+1} \cdot \binom{i_1}{j_1} \cdot \ldots \cdot (-1)^{j_{n-1}+1} \cdot \binom{i_{n-1}}{j_{n-1}} \\ &= (-1)^{j_0+j_1+\cdots+j_{n-1}+n} \cdot \binom{i_0}{j_0} \cdot \binom{i_1}{j_1} \cdot \ldots \cdot \binom{i_{n-1}}{j_{n-1}}.\end{aligned}$$

By a theorem of Lucas ([5], see also [2]), if $h$ is a prime, then the product of binomial coefficients in the above formula is congruent to $\binom{i}{j}$ modulo $h$. Thus, we have

$$\left(T_h^{\otimes n}\right)_{ij} = (-1)^n \cdot (-1)^{j_0+j_1+\cdots+j_{n-1}} \cdot \binom{i}{j}.$$

Now if $h$ is odd, then $j = j_0 + j_1 h + \cdots + j_{n-1}h^{n-1} \equiv j_0 + j_1 + \cdots + j_{n-1} \pmod 2$, hence $\left(T_h^{\otimes n}\right)_{ij} = (-1)^n \cdot (-1)^j \cdot \binom{i}{j} = (-1)^n \cdot p_{ij}$, as claimed. If $h = 2$, then $1 \equiv -1 \pmod h$, so the signs do not matter at all in this case, hence $\left(T_h^{\otimes n}\right)_{ij} = \binom{i}{j} = (-1)^n \cdot p_{ij}$. □

We will study the number of eigenvectors of the Pascal and Reed-Muller-Fourier transforms, and, more generally of self-inverse triangular matrices. If $S \in \mathbb{Z}_h^{N\times N}$ is a self-inverse matrix and $\mathbf{0} \neq \mathbf{u} \in \mathbb{Z}_h^N$ is an eigenvector of $S$ corresponding to the eigenvalue $\lambda \in \mathbb{Z}_h$, then $\mathbf{u} = S^2\mathbf{u} = \lambda S\mathbf{u} = \lambda^2\mathbf{u}$. Now if $h$ is a prime number, then this implies that $\lambda^2 = 1$. As the next example shows, if $h$ is a composite number, then there might be eigenvalues $\lambda$ such that $\lambda^2 \neq 1$.

**Example 2.1.** The eigenspace $U_3 \leq \mathbb{Z}_6^6$ of the matrix $T_6$ corresponding to the eigenvalue $\lambda = 3$ is

$$U_3 = \{(0, a, a, b, a, c) : a, b, c \in \{0, 3\}\}.$$

This eigenspace has 8 elements, which is not a power of $h = 6$, hence $U_3$ does not have a basis.

One can see other examples in Table 8, which lists the sizes of the eigenspaces of $T_h$ for $h \leq 12$. In contrast, we will consider only $\lambda$ eigenvalues with $\lambda^2 = 1$. This certainly includes the cases $\lambda = 1$ (fixed points) and $\lambda = -1$, but in general there might be more such eigenvalues (for example, if $h = 12$, then $\lambda = 5$ and $\lambda = 7$ also satisfy $\lambda^2 = 1$). It was proved in [18] that if $h$ is odd, then $\mathbb{Z}_h^{h^n}$ has a basis consisting of eigenvectors of $T_h^{\otimes n}$ corresponding to the eigenvalues 1 and $-1$. If $h$ is a prime (i.e., if $\mathbb{Z}_h$ is a field), then this implies that there are no other eigenvalues. However, as we can see in Table 8, if $h$ is a composite number, then this is not true: for $h = 9$ there exists eigenvectors corresponding to $\lambda = 2$, 4, 5, 7.

# 3 Triangular self-inverse transforms over domains of odd size

If $h$ is odd and $S \in \mathbb{Z}_h^{N\times N}$ is a triangular self-inverse matrix, then we can get a quite general formula for the number of eigenvectors of $S$ corresponding to an eigenvalue $\lambda \in \mathbb{Z}_h$ with $\lambda^2 = 1$. Actually, the size of the eigenspace depends only on the diagonal entries of $S$ (and, of course, on $h$ and $\lambda$ as well). The key observation is that $\mathbb{Z}_h^N$ is the direct sum of the subspaces $U_\lambda$ and $U_{-\lambda}$.

**Lemma 3.1.** *Assume that $h$ is odd and $S$ is an $N \times N$ matrix over $\mathbb{Z}_h$ such that $S^2 = I_N$. If $\lambda \in \mathbb{Z}_h$ and $\lambda^2 = 1$, then $\mathbb{Z}_h^N$ is the direct sum of the eigenspaces of $S$ corresponding to the eigenvalues $\lambda$ and $-\lambda$, i.e., $\mathbb{Z}_h^N = U_\lambda \oplus U_{-\lambda}$.*

*Proof.* For arbitrary $\mathbf{v} \in \mathbb{Z}_h^N$, let $\mathbf{v}^+ = \frac{1}{2}(\mathbf{v} + \lambda S\mathbf{v})$ and $\mathbf{v}^- = \frac{1}{2}(\mathbf{v} - \lambda S\mathbf{v})$. Note that these expressions are well defined, because $h$ is odd, thus 2 has a multiplicative inverse in $\mathbb{Z}_h$. Clearly, we have $\mathbf{v} = \mathbf{v}^+ + \mathbf{v}^-$; moreover, $\mathbf{v}^+ \in U_\lambda$ and $\mathbf{v}^- \in U_{-\lambda}$ follow from the fact that $S^2 = I_N$ and $\lambda^2 = 1$:

$$S\mathbf{v}^+ = \frac{1}{2}\left(S\mathbf{v} + \lambda S^2\mathbf{v}\right) = \frac{1}{2}\left(\lambda^2 S\mathbf{v} + \lambda\mathbf{v}\right) = \lambda\mathbf{v}^+;$$
$$S\mathbf{v}^- = \frac{1}{2}\left(S\mathbf{v} - \lambda S^2\mathbf{v}\right) = \frac{1}{2}\left(\lambda^2 S\mathbf{v} - \lambda\mathbf{v}\right) = -\lambda\mathbf{v}^-.$$

This means that $\mathbb{Z}_h^N = U_\lambda + U_{-\lambda}$. It remains to be proved that $U_\lambda \cap U_{-\lambda} = \{\mathbf{0}\}$. If $\mathbf{u} \in U_\lambda \cap U_{-\lambda}$, then $S\mathbf{u} = \lambda\mathbf{u} = -\lambda\mathbf{u}$, hence $2\lambda\mathbf{u} = \mathbf{0}$. Since $\lambda^2 \equiv 1 \pmod{h}$, we have $\gcd(h, \lambda) = 1$; moreover, 2 is also relatively prime to $h$, as $h$ is odd. Therefore we may conclude that $\mathbf{u} = \mathbf{0}$, and this completes the proof. □

We still need a simple number-theoretical lemma before we can prove our main theorem about the number of eigenvectors.

**Lemma 3.2.** *If $h$ is an odd natural number, and $\lambda, s \in \mathbb{Z}$ are such that $\lambda^2 \equiv s^2 \equiv 1 \pmod{h}$, then $\gcd(h, s-\lambda) \cdot \gcd(h, s+\lambda) = h$.*

*Proof.* Let $h = \prod p_i^{e_i}$ be the prime power factorization of $h$, where each $p_i$ is an odd prime and each $e_i$ is a positive exponent. Since $\lambda^2 \equiv 1 \pmod{h}$, we have $p_i^{e_i} \mid (\lambda - 1)(\lambda + 1)$ for every $i$. This implies that either $p_i^{e_i} \mid \lambda - 1$ or $p_i^{e_i} \mid \lambda + 1$, as $\gcd(\lambda - 1, \lambda + 1) \leq 2$ and $p_i$ is odd. Thus $\lambda \equiv \pm 1 \pmod{p_i^{e_i}}$, and a similar argument shows that $s \equiv \pm 1 \pmod{p_i^{e_i}}$ for every $i$. Therefore, one of $s - \lambda$ and $s + \lambda$ is congruent to $\pm 2$ and the other one is congruent to 0 modulo $p_i^{e_i}$. Thus one of $\gcd(h, s-\lambda)$ and $\gcd(h, s+\lambda)$ is divisible by $p_i^{e_i}$ and the other one is not divisible by $p_i$. This is true for every prime divisor $p_i$ of $h$, and no other primes can occur as a divisor of $\gcd(h, s-\lambda) \cdot \gcd(h, s+\lambda)$, hence we may conclude that $\gcd(h, s-\lambda) \cdot \gcd(h, s+\lambda) = \prod p_i^{e_i} = h$. □

**Theorem 3.1.** *Assume that $h$ is odd and $S = (s_{ij})_{i,j=0}^{N-1}$ is a lower triangular $N \times N$ matrix over $\mathbb{Z}_h$ such that $S^2 = I_N$. If $\lambda \in \mathbb{Z}_h$ and $\lambda^2 = 1$, then the size of the eigenspace $U_\lambda(S)$ of $S$ corresponding to the eigenvalue $\lambda$ is*

$$|U_\lambda(S)| = \gcd(h, s_{00} - \lambda) \cdot \ldots \cdot \gcd(h, s_{N-1,N-1} - \lambda).$$

*Proof.* The elements of $U_\lambda$ are the solutions of the system $(S - \lambda I_N)\mathbf{x} = \mathbf{0}$ of homogeneous linear equations. The first equation (written as a modulo $h$ congruence) is $(s_{00} - \lambda)x_0 \equiv 0 \pmod{h}$. This linear congruence has $\gcd(h, s_{00} - \lambda)$ many solutions modulo $h$, thus there are $\gcd(h, s_{00} - \lambda)$ possible values for $x_0 \in \mathbb{Z}_h$. The second equation is equivalent to $s_{10}x_0 + (s_{11} - \lambda)x_1 \equiv 0 \pmod{h}$. If we have

already chosen the value of $x_0$, then this can be viewed as a linear congruence $(s_{11}-\lambda)\,x_1 \equiv -s_{10}x_0 \pmod h$ for the unknown $x_1$. Depending on the value of $x_0$, this linear congruence may or may not have a solution, but if there is a solution, then the number of solutions modulo $h$ is $\gcd(h, s_{11}-\lambda)$. Thus the number of choices for $x_1 \in \mathbb{Z}_h$ is either 0 or $\gcd(h, s_{11}-\lambda)$. Continuing in this manner, having assigned values to $x_0, \dots, x_{i-1}$, we can treat the $i$-th equation as a linear congruence $(s_{ii}-\lambda)\,x_i \equiv -s_{i0}x_0 - \dots - s_{i,i-1}x_{i-1} \pmod h$ for the unknown $x_i$, which has either 0 or $\gcd(h, s_{ii}-\lambda)$ many solutions in $\mathbb{Z}_h$. This provides an upper estimate for the size of the eigenspace $U_\lambda$:

$$|U_\lambda| \le \gcd(h, s_{00}-\lambda) \cdot \ldots \cdot \gcd(h, s_{N-1,N-1}-\lambda)\,. \tag{3}$$

Let us write down the corresponding estimate for $-\lambda$, and use Lemma 3.2 (observe that $S^2 = I_N$ implies that $s_{ii}^2 = 1$ for every $i$, since $S$ is a lower triangular matrix):

$$|U_\lambda| \cdot |U_{-\lambda}| \le \gcd(h, s_{00}-\lambda)\gcd(h, s_{00}+\lambda) \cdot \ldots \\ \cdot \gcd(h, s_{N-1,N-1}-\lambda)\gcd(h, s_{N-1,N-1}+\lambda) = h^N.$$

By Lemma 3.1, every element of $\mathbb{Z}_h^N$ can be uniquely expressed as a sum of a vector from $U_\lambda$ and a vector from $U_{-\lambda}$. This implies that $|U_\lambda| \cdot |U_{-\lambda}| = \left|\mathbb{Z}_h^N\right| = h^N$, hence the inequality above is in fact an equality, so we have equality in (3) as well. □

# 4 The Pascal transform

Next, we will determine the number of eigenvectors of $P_N$ corresponding to eigenvalues $\lambda \in \mathbb{Z}_h$ with $\lambda^2 = 1$ (note that Theorem 4.1, the main result of this section, overlaps with Theorem 3.1 if $h$ is odd). Since $T_h = -P_h$, this includes as a special case the results of [18], where one-variable eigenfunctions of the Reed-Muller-Fourier transform were considered with the eigenvalues $\pm 1$. An elimination procedure was used in [18], but its correctness was not rigorously proved (although the patterns of binomial coefficients appearing in the matrices were clear enough). Here we provide a proof, and instead of a step-by-step procedure, we do the elimination at once, by multiplying by a suitable invertible matrix.

Let $A_N = (a_{ij})_{i,j=0}^{N-1} \in \mathbb{Z}_h^{N\times N}$ be the matrix given by the entries

$$a_{ij} = (-1)^{i+j} \cdot \binom{\lfloor i/2 \rfloor}{i-j}.$$

As an example, the matrix $A_8$ is shown in Table 6. We will determine the number of solutions of $(P_N - \lambda I_N)\,\mathbf{x} = \mathbf{0}$ by multiplying by $A_N$ on the left. The following combinatorial identity is required to compute the product $A_N P_N$. Such identities can be proved automatically by a computer [10], but a "human" proof might still be of interest.

**Lemma 4.1.** *For all natural numbers $\ell, r$ and $m$, we have*

$$\sum_{k=0}^{r} (-1)^k \cdot \binom{r}{k} \cdot \binom{\ell + r - k}{m} \;=\; \binom{\ell}{m-r}. \tag{4}$$

*Proof.* We give a combinatorial interpretation of the identity, and, to make the proof more vivid, we present it in the setting of a fantasy story. Assume that there is a group of $r$ orcs and $\ell$ e$\ell$ves wandering together in Middle-earth. They learn about a wizard forging magic rings, and they decide to steal some of those rings. A set of $m$ members of the group is to be chosen for this mission, such that all the orcs are included (they are good fighters). Thus it suffices to choose the $m-r$ elves that are going with the orcs, and the number of such choices is obviously $\binom{\ell}{m-r}$.

Now we count the number of possibilities once more, with the help of the inclusion-exclusion principle, and this will result in the left hand side of (4). Let $E$ and $O$ denote the set of elves and orcs (thus $|E| = \ell$ and $|O| = r$), and let $\mathcal{G}$ stand for the set of "good" choices for the mission:

$$\mathcal{G} = \{M \subseteq E \cup O : |M| = m \text{ and } O \subseteq M\}.$$

We saw in the previous paragraph that $|\mathcal{G}| = \binom{\ell}{m-r}$. For every orc $o \in O$, let $\mathcal{B}_o$ denote the set of choices that are "bad", because the orc $o$ is not sent to the mission:

$$\mathcal{B}_o = \{M \subseteq E \cup O : |M| = m \text{ and } o \notin M\}.$$

Given $k$ orcs $o_1, \ldots, o_k \in O$, the cardinality of $\mathcal{B}_{o_1} \cap \cdots \cap \mathcal{B}_{o_k}$ is $\binom{\ell+r-k}{m}$, and there are $\binom{r}{k}$ possibilities for the set $\{o_1, \ldots, o_k\}$. Therefore, by the inclusion-exclusion principle, we have

$$|\mathcal{G}| = \sum_{k=0}^{r} (-1)^k \cdot \binom{r}{k} \cdot \binom{\ell + r - k}{m},$$

which is indeed the left hand side of (4). □

**Lemma 4.2.** *The entries of the matrix $A_N P_N$ are the following:*

$$(A_N P_N)_{ij} = (-1)^j \cdot \binom{\lceil i/2 \rceil}{i - j} \qquad (i, j = 0, 1, \ldots, N-1).$$

*Proof.* From the definitions of the matrices $A_N$ and $P_N$, we have

$$\begin{aligned}(A_N P_N)_{ij} = \sum_{k=0}^{N-1} a_{ik} \cdot p_{kj} &= \sum_{k=0}^{N-1} (-1)^{i+k} \cdot \binom{\lfloor i/2 \rfloor}{i-k} \cdot (-1)^j \cdot \binom{k}{j} \\ &= (-1)^j \cdot \sum_{k=0}^{\lfloor i/2 \rfloor} (-1)^k \cdot \binom{\lfloor i/2 \rfloor}{k} \cdot \binom{i-k}{j}.\end{aligned}$$

(In the last step we changed the summation variable from $k$ to $i-k$, and we omitted those terms where the first binomial coefficient is zero.) Applying Lemma 4.1 with $r = \lfloor i/2 \rfloor$, $\ell = \lceil i/2 \rceil$ and $m = j$, we get $(-1)^j \cdot \binom{\lceil i/2 \rceil}{j - \lfloor i/2 \rfloor} = (-1)^j \cdot \binom{\lceil i/2 \rceil}{i-j}$, hence the lemma is proved. □

**Theorem 4.1.** *For every natural number $h$ and $\lambda \in \mathbb{Z}_h$ with $\lambda^2 = 1$, the eigenspace $U_\lambda(P_N) \leq \mathbb{Z}_h^N$ of the discrete Pascal transform $P_N$ has cardinality*

$$|U_\lambda(P_N)| = \begin{cases} h^{\lfloor N/2 \rfloor} \cdot \gcd(1-\lambda, h), & \text{if } N \text{ is odd;} \\ h^{N/2}, & \text{if } N \text{ is even.} \end{cases}$$

*Proof.* We need to determine the set of vectors $\mathbf{x} \in \mathbb{Z}_h^N$ satisfying $(P_N - \lambda I_N)\mathbf{x} = \mathbf{0}$. Since the matrix $A_N$ is triangular and all of its entries on the main diagonal are 1, we have $\det(A_N) = 1$, hence $A_N$ has an inverse in $\mathbb{Z}_h^{N\times N}$. Therefore, the solutions of $(P_N - \lambda I_N)\mathbf{x} = \mathbf{0}$ are the same as the solutions of $A_N(P_N - \lambda I_N)\mathbf{x} = \mathbf{0}$. We will prove that we can omit (roughly) every second equation from this system of linear equations: row $i$ of the matrix $A_N(P_N - \lambda I_N) = A_N P_N - \lambda A_N$ is a scalar multiple of row $i+1$ whenever $i$ is even and $i < N-1$.

Letting $i = 2k$, the $j$-th entries of row $i$ and of row $i+1$ are, by Lemma 4.2 and by the definition of the matrix $A_N$,

$$(A_N P_N - \lambda A_N)_{2k,j} = (-1)^j \cdot (1-\lambda) \cdot \binom{k}{2k-j}, \tag{5a}$$

$$(A_N P_N - \lambda A_N)_{2k+1,j} = (-1)^j \cdot \left( \binom{k+1}{2k+1-j} + \lambda \cdot \binom{k}{2k+1-j} \right). \tag{5b}$$

Multiplying (5b) by $1-\lambda$ and taking into account the fact that $\lambda^2 = 1$ (and also using the usual recurrence for the Pascal triangle), we indeed get (5a):

$$\begin{aligned} (1-\lambda) \cdot (A_N P_N - \lambda A_N)_{2k+1,j} = \\ &= (-1)^j \cdot \left( (1-\lambda) \cdot \binom{k+1}{2k+1-j} + (\lambda - \lambda^2) \cdot \binom{k}{2k+1-j} \right) \\ &= (-1)^j \cdot (1-\lambda) \cdot \left( \binom{k+1}{2k+1-j} - \binom{k}{2k+1-j} \right) \\ &= (-1)^j \cdot (1-\lambda) \cdot \binom{k}{2k-j} \\ &= (A_N P_N - \lambda A_N)_{2k,j}. \end{aligned}$$

Therefore, the (equations corresponding to the) even-numbered rows can be omitted without changing the set of solutions. Let us distinguish two cases based on the parity of $N$.

If $N$ is even, then we keep row $i$ for $i = 1, 3, \ldots, N-1$. From (5b) we see that the first nonzero entry in row $2k+1$ is $(A_N P_N - \lambda A_N)_{2k+1,k} = (-1)^k$. Therefore, after deleting the even-numbered rows, we get an $(N/2) \times N$ matrix with the following form:

$$\begin{pmatrix} 1 & * & \cdots & * & * & \cdots & * \\ 0 & -1 & \cdots & * & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & (-1)^{N/2-1} & * & \cdots & * \end{pmatrix}.$$

This matrix is in row echelon form, hence we can see that in the corresponding system of linear equations the last $N/2$ variables (namely $x_{N/2}, \ldots, x_{N-1}$) are free, and the first $N/2$ variables (namely $x_0, \ldots, x_{N/2-1}$) can be uniquely determined from the free variables. Since we have $h$ choices for each of the free variables $x_{N/2}, \ldots, x_{N-1}$, the cardinality of $U_\lambda$ is $h^{N/2}$.

Now let us assume that $N$ is odd. In this case we cannot delete row $N-1$ even though $N-1$ is even, because this is the last row in the matrix (hence it cannot be a scalar multiple of the next row, as the next row does not exist). Thus we keep row $i$ for $i = 1, 3, \ldots, N-2, N-1$, hence we get an $\lceil N/2 \rceil \times N$ matrix. Computing the first nonzero entry in each row with the help of (5a) and (5b), we see that our matrix has the following form:

$$\begin{pmatrix} 1 & * & \cdots & * & * & * & \cdots & * \\ 0 & -1 & \cdots & * & * & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & (-1)^{(N-3)/2} & * & * & \cdots & * \\ 0 & 0 & \cdots & 0 & (-1)^{(N-1)/2} \cdot (1-\lambda) & * & \cdots & * \end{pmatrix}.$$

By (5a), each element in the last row in the above matrix (row $N-1$ in the original matrix before deleting every second row) has a factor $1-\lambda$. Thus the last row can be divided by $1-\lambda$, but then we obtain a modulo $h/\gcd(1-\lambda, h)$ congruence (instead of a modulo $h$ congruence). Therefore, $x_{\lfloor N/2 \rfloor}$ is determined by the free variables $x_{\lceil N/2 \rceil}, \ldots, x_{N-1}$ only modulo $\gcd(1-\lambda, h)$, so there are $\gcd(1-\lambda, h)$ possibilities for $x_{\lfloor N/2 \rfloor}$ in $\mathbb{Z}_h$. The variables $x_0, \ldots, x_{\lfloor N/2 \rfloor - 1}$ are then uniquely determined (modulo $h$). We may conclude that the number of solutions is $h^{\lfloor N/2 \rfloor} \cdot \gcd(1-\lambda, h)$. □

**Corollary 4.1.** *For all natural numbers $h \geq 2$ and $n \geq 1$, the number of fixed points of the discrete Pascal transform $P_N$ on $\mathbb{Z}_h^N$ is $h^{\lceil N/2 \rceil}$.*

*Proof.* We just need to apply Theorem 4.1 with $\lambda = 1$ and note that if $N$ is odd, then $|U_1| = h^{\lfloor N/2 \rfloor} \cdot \gcd(1-\lambda, h) = h^{\lfloor N/2 \rfloor} \cdot \gcd(0, h) = h^{\lfloor N/2 \rfloor} \cdot h = h^{\lceil N/2 \rceil}$. □

# 5 The Reed-Muller-Fourier transform

If $h$ is odd, then the results of Section 3 apply to the Reed-Muller-Fourier transform. By Proposition 2.1, Section 4 also covers the Reed-Muller-Fourier transform when $h$ is a prime number.

From now on, we will assume that $h$ is even, and we consider eigenvectors of $T_h^{\otimes n}$ corresponding to an eigenvalue $\lambda \in \mathbb{Z}_h$ such that $\lambda^2 = 1$. (Note that this implies that $\lambda$ is odd and relatively prime to $h$.) In this case $\mathbb{Z}_h^{h^n}$ is not the direct sum of the eigenspaces $U_\lambda$ and $U_{-\lambda}$, but we can still determine the cardinalities of $U_\lambda + U_{-\lambda}$ and $U_\lambda \cap U_{-\lambda}$ (see Lemma 5.2 and Lemma 5.3).

**Lemma 5.1.** *If $h$ is an even natural number, then the number of vectors $\mathbf{u} \in \mathbb{Z}_2^{h^n}$ satisfying $T_h^{\otimes n}\mathbf{u} \equiv \mathbf{u} \pmod 2$ is $2^{h^n/2}$.*

*Proof.* Let us replace each entry of $T_h$ by its residue modulo 2, and let $B_h \in \mathbb{Z}_2^{h\times h}$ denote the resulting matrix over $\mathbb{Z}_2$. Then $T_h^{\otimes n} \equiv B_h^{\otimes n} \pmod 2$, and our task is to prove that $\dim U_1\left(B_h^{\otimes n}\right) = h^n/2$. Since $B_h^{\otimes n}$ is a lower triangular matrix with ones on its diagonal, we can use Lemma 2.1 repeatedly to prove that

$$\dim U_1\left(B_h^{\otimes n}\right) \le h^{n-1} \cdot \dim U_1\left(B_h\right). \tag{6}$$

Note that $B_h$ is none other than $P_h$ taken modulo 2, hence applying Corollary 4.1 (substituting $N$ with $h$ and $h$ with 2), we see that the number of fixed points of $B_h$ is $2^{h/2}$. This means that $\dim U_1\left(B_h\right) = h/2$, and then (6) gives $\dim U_1\left(B_h^{\otimes n}\right) \le h^n/2$.

To prove the reverse inequality, observe that $\left(B_h^{\otimes n} - I_{h^n}\right)^2 = \left(B_h^{\otimes n}\right)^2 - 2B_h^{\otimes n} + I_{h^n} = 0_{h^n}$, since $B_h^{\otimes n}$ is a self-inverse matrix and the matrices are considered modulo 2. This implies that the range of $B_h^{\otimes n} - I_{h^n}$ is contained in its kernel, hence $\operatorname{rank}\left(B_h^{\otimes n} - I_{h^n}\right) \le \dim\ker\left(B_h^{\otimes n} - I_{h^n}\right)$. By the rank-nullity theorem, we have

$$\begin{aligned} h^n &= \operatorname{rank}\left(B_h^{\otimes n} - I_{h^n}\right) + \dim\ker\left(B_h^{\otimes n} - I_{h^n}\right) \\ &\le 2 \cdot \dim\ker\left(B_h^{\otimes n} - I_{h^n}\right) = 2 \cdot \dim U_1\left(B_h^{\otimes n}\right), \end{aligned}$$

and this proves that $\dim U_1\left(B_h^{\otimes n}\right) \ge h^n/2$. □

**Lemma 5.2.** *If $h$ is an even natural number, $\lambda \in \mathbb{Z}_h$ and $\lambda^2 = 1$, then the cardinality of the sum of the eigenspaces $U_\lambda, U_{-\lambda} \le \mathbb{Z}_h^{h^n}$ of $T_h^{\otimes n}$ is*

$$|U_\lambda + U_{-\lambda}| = \frac{h^{h^n}}{2^{h^n/2}}.$$

*Proof.* We claim that

$$U_\lambda + U_{-\lambda} = \left\{\mathbf{v} \in \mathbb{Z}_h^{h^n} : T_h^{\otimes n}\mathbf{v} \equiv \mathbf{v} \pmod 2\right\}. \tag{7}$$

If $\mathbf{v} = \mathbf{v}^+ + \mathbf{v}^-$ with $\mathbf{v}^+ \in U_\lambda, \mathbf{v}^- \in U_{-\lambda}$, then $T_h^{\otimes n}\mathbf{v} = \lambda\mathbf{v}^+ - \lambda\mathbf{v}^- \equiv \mathbf{v}^+ + \mathbf{v}^- \equiv \mathbf{v} \pmod 2$, as $\lambda$ is odd. Now assume that $T_h^{\otimes n}\mathbf{v} \equiv \mathbf{v} \pmod 2$. Then each entry of $\mathbf{v} + \lambda T_h^{\otimes n}\mathbf{v}$ is even (again, we make use of the fact that $\lambda$ is odd), hence it makes sense to write $\mathbf{v}^+ = \frac{1}{2}\left(\mathbf{v} + \lambda T_h^{\otimes n}\mathbf{v}\right)$. Similarly, we can let $\mathbf{v}^- = \frac{1}{2}\left(\mathbf{v} - \lambda T_h^{\otimes n}\mathbf{v}\right)$. It is clear that $\mathbf{v} = \mathbf{v}^+ + \mathbf{v}^-$, and the same argument as in the proof of Lemma 3.1 shows that $\mathbf{v}^+ \in U_\lambda$ and $\mathbf{v}^- \in U_{-\lambda}$. Therefore, $\mathbf{v} \in U_\lambda + U_{-\lambda}$, and this proves (7).

The above arguments show that we need to count the vectors $\mathbf{v} \in \mathbb{Z}_h^{h^n}$ for which there exists some $\mathbf{u} \in \mathbb{Z}_2^{h^n}$ such that $T_h^{\otimes n}\mathbf{u} \equiv \mathbf{u} \pmod 2$ and $\mathbf{v} \equiv \mathbf{u} \pmod 2$. By Lemma 5.1, there are $2^{h^n/2}$ possibilities for $\mathbf{u}$. Once $\mathbf{u}$ us given, we have $(h/2)^{h^n}$ choices for $\mathbf{v}$: if $u_i = 0$, then $v_i \in \{0, 2, \dots, h\}$, and if $u_i = 1$, then $v_i \in \{1, 3, \dots, h-1\}$ for $i = 1, 2, \dots, h^n$. We may conclude that the number of $\mathbf{v} \in \mathbb{Z}_h^{h^n}$ with $T_h^{\otimes n}\mathbf{v} \equiv \mathbf{v} \pmod 2$ is $2^{h^n/2} \cdot (h/2)^{h^n}$, and this completes the proof. □

**Lemma 5.3.** *If $h$ is an even natural number, $\lambda \in \mathbb{Z}_h$ and $\lambda^2 = 1$, then the cardinality of the intersection of the eigenspaces $U_\lambda, U_{-\lambda} \leq \mathbb{Z}_h^{h^n}$ of $T_h^{\otimes n}$ is*

$$|U_\lambda \cap U_{-\lambda}| = 2^{h^n/2}.$$

*Proof.* We claim that

$$U_\lambda \cap U_{-\lambda} = \Big\{ \mathbf{v} \in \mathbb{Z}_h^{h^n} : \exists \mathbf{u} \in \mathbb{Z}_2^{h^n} \text{ such that } \mathbf{v} = \frac{h}{2} \cdot \mathbf{u} \text{ and } T_h^{\otimes n}\mathbf{u} \equiv \mathbf{u} \pmod 2 \Big\}. \tag{8}$$

If $\mathbf{v} \in U_\lambda \cap U_{-\lambda}$, then $T_h^{\otimes n}\mathbf{v} = \lambda\mathbf{v} = -\lambda\mathbf{v}$, hence $2\lambda\mathbf{v} = \mathbf{0}$. Since $\lambda$ is relatively prime to $h$, the condition $2\lambda\mathbf{v} = \mathbf{0}$ is equivalent to $\mathbf{v} \equiv \mathbf{0} \pmod{h/2}$, i.e., each component of $\mathbf{v}$ is either 0 or $h/2$. Therefore, $\mathbf{v}$ can be written as $h/2 \cdot \mathbf{u}$, where $u_i = 0$ if $v_i = 0$ and $u_i = 1$ if $v_i = h/2$. Now $T_h^{\otimes n}\mathbf{v} = \lambda\mathbf{v}$ can be reformulated as $h/2 \cdot T_h^{\otimes n}\mathbf{u} = h/2 \cdot \lambda\mathbf{u}$, which is equivalent to $T_h^{\otimes n}\mathbf{u} \equiv \lambda\mathbf{u} \equiv \mathbf{u} \pmod 2$, as $\lambda$ is odd. Next, assume that $\mathbf{v} = h/2 \cdot \mathbf{u}$ for some $\mathbf{u} \in \mathbb{Z}_2^{h^n}$ such that $T_h^{\otimes n}\mathbf{u} \equiv \mathbf{u} \pmod 2$. Then we have $T_h^{\otimes n}\mathbf{v} = h/2 \cdot T_h^{\otimes n}\mathbf{u}$; furthermore, $T_h^{\otimes n}\mathbf{u} \equiv \mathbf{u} \pmod 2$ implies that $h/2 \cdot T_h^{\otimes n}\mathbf{u} \equiv h/2 \cdot (\pm\lambda\mathbf{u}) \pmod h$, since $\lambda$ is odd. Thus we have $T_h^{\otimes n}\mathbf{v} \equiv h/2 \cdot (\pm\lambda\mathbf{u}) \equiv \pm\lambda\mathbf{v} \pmod h$, and this proves (8).

Since $\mathbf{v}$ is uniquely determined by $\mathbf{u}$ in (8), we may conclude that $|U_\lambda \cap U_{-\lambda}| = \big|\{\mathbf{u} \in \mathbb{Z}_2^{h^n} : T_h^{\otimes n}\mathbf{u} \equiv \mathbf{u} \pmod 2\}\big|$, and this is $2^{h^n/2}$ by Lemma 5.1. □

Lemmas 5.2 and 5.3 allow us to determine the product $|U_\lambda| \cdot |U_{-\lambda}|$ (see the first paragraph of the proof of Theorem 5.1). This will give the cardinalities of the eigenspaces if we manage to prove that $|U_\lambda| = |U_{-\lambda}|$. To achieve this, we use an auxiliary matrix $C_h = (c_{ij})_{i,j=0}^{h-1} \in \mathbb{Z}_h^{h\times h}$ given by

$$c_{ij} = (-1)^{j+1} \cdot 2^{j-i} \cdot \binom{h-1-i}{j-i}.$$

As an illustration, see Table 7, which shows this matrix for $h = 8$. Just like that with the matrix $A_h$ in Section 4, a combinatorial identity is required to compute the products $C_hT_h$ and $T_hC_h$. It should be mentioned that the algorithms of [10] tell us that the sums in (9) below do not have a closed form.

**Lemma 5.4.** *For all natural numbers $\ell, r$ and $m$, we have*

$$\sum_{k=0}^{\ell} (-1)^k \cdot \binom{\ell}{k} \cdot \binom{\ell+r-k}{m} \cdot 2^{\ell-k} \;=\; \sum_{k=0}^{r} (-1)^k \cdot \binom{r}{k} \cdot \binom{\ell+r-k}{m-k} \cdot 2^{m-k}. \tag{9}$$

*Proof.* Let us visit the elves and orcs of Lemma 4.1 once more. They managed to fetch a generous supply of magic rings; in principle, each member of the group could wear one. However, such artefacts can be dangerous, so they should be used with care. Therefore, when a set $M$ of $m$ members of the group are chosen for the next adventure, some rules must be observed regarding the set $R$ of ring-bearers. First, orcs should not wear magic rings, because they do not have the mental skills

required to handle them safely. Second, those staying at home should not wear magic rings, since they will not need them. We will prove that both sides of (9) give the cardinality of the following set of good assignments:

$$\mathcal{G} = \{(M,R) : M, R \subseteq E \cup O,\ |M| = m \text{ and } R \subseteq E \cap M\}.$$

We will use the inclusion-exclusion principle in two different ways to count the elements of $\mathcal{G}$. Let us spell(!) out the requirements on the pair $(M,R)$ in detail:

(i) if $e \in E \setminus M$, then $e \notin R$;

(ii) if $o \in O \cap M$, then $o \notin R$;

(iii) if $o \in O \setminus M$, then $o \notin R$.

First, let $\mathcal{B}_e$ denote the set of assignments where conditions (ii) and (iii) are satisfied but (i) is not, because an elf $e \in E$ gets a ring, even though (s)he stays at home:

$$\mathcal{B}_e = \{(M,R) : M, R \subseteq E \cup O,\ |M| = m \text{ and } e \in R \subseteq E\}.$$

Given $k$ elves $e_1, \dots, e_k \in E$, the cardinality of $\mathcal{B}_{e_1} \cap \dots \cap \mathcal{B}_{e_k}$ is $\binom{\ell+r-k}{m} \cdot 2^{\ell-k}$. Indeed, there are $\binom{\ell+r-k}{m}$ possibilities for $M$, as $e_1, \dots, e_k \notin M$, and we can distribute the rings to the elves (other than $e_1, \dots, e_k$, who already received their rings) in $2^{\ell-k}$ many ways. There are $\binom{\ell}{k}$ options for the set $\{e_1, \dots, e_k\}$, thus the inclusion-exclusion principle gives the left hand side of (9) for $|\mathcal{G}|$.

Now let $\mathcal{C}_e$ denote the set of assignments where the requirements (i) and (iii) are met but (ii) is violated, because an orc $o \in O$ taking part in the mission gets a ring:

$$\mathcal{C}_e = \{(M,R) : M, R \subseteq E \cup O,\ |M| = m \text{ and } o \in R \subseteq M\}.$$

Given $k$ orcs $o_1, \dots, o_k \in O$, the cardinality of $\mathcal{C}_{o_1} \cap \dots \cap \mathcal{C}_{o_k}$ is $\binom{\ell+r-k}{m-k} \cdot 2^{m-k}$: we have $\binom{\ell+r-k}{m-k}$ many options to choose those members of $E \cup O$ that will accompany $o_1, \dots, o_k$ on the mission, and we can distribute the rings to the members of $M$ (other than $o_1, \dots, o_k$, who have already received their rings) in $2^{m-k}$ many ways. There are $\binom{r}{k}$ choices for the set $\{o_1, \dots, o_k\}$, so the inclusion-exclusion principle indeed gives the right hand side of (9) for $|\mathcal{G}|$. □

**Lemma 5.5.** *If $h$ is an even natural number, then $T_h C_h = -C_h T_h$.*

*Proof.* Let us compute first the entries of $T_h C_h$ (in the last step we omit terms where the first binomial coefficient is zero):

$$\begin{aligned}(T_h C_h)_{ij} = \sum_{k=0}^{h-1} t_{ik} \cdot c_{kj} &= \sum_{k=0}^{h-1} (-1)^{k+1} \cdot \binom{i}{k} \cdot (-1)^{j+1} \cdot 2^{j-k} \cdot \binom{h-1-k}{j-k} \\ &= (-1)^j \cdot \sum_{k=0}^{i} (-1)^k \cdot \binom{i}{k} \cdot \binom{h-1-k}{j-k} \cdot 2^{j-k}.\end{aligned}$$

This is the same as $(-1)^j$ times the right hand side of (9) with $r = i$, $\ell = h-1-i$ and $m = j$. Similarly, for $C_h T_h$ we find that

$$\begin{aligned}(C_h T_h)_{ij} = \sum_{g=0}^{h-1} c_{ig} \cdot t_{gj} &= \sum_{g=0}^{h-1} (-1)^{g+1} \cdot 2^{g-i} \cdot \binom{h-1-i}{g-i} \cdot (-1)^{j+1} \cdot \binom{g}{j} \\ &= \sum_{g=i}^{h-1} (-1)^{g+j} \cdot \binom{h-1-i}{g-i} \cdot \binom{g}{j} \cdot 2^{g-i}.\end{aligned}$$

Now let us introduce a new summation variable $k = h-1-g$:

$$(-1)^{h-1+j} \cdot \sum_{k=0}^{h-1-i} (-1)^k \cdot \binom{h-1-i}{k} \cdot \binom{h-1-k}{j} \cdot 2^{h-1-i-k}.$$

With the same setting for $r$, $\ell$ and $m$ as above, this becomes $(-1)^{h-1+j}$ times the left hand side of (9). Therefore, Lemma 5.4 implies that $(-1)^j \cdot (T_h C_h)_{ij} = (-1)^{h-1+j} \cdot (C_h T_h)_{ij}$. If $h$ is even, then $(-1)^j$ and $(-1)^{h-1+j}$ are of opposite sign, hence $(T_h C_h)_{ij} = -(C_h T_h)_{ij}$. □

Lemma 5.5 allows us to give a bijection between $U_\lambda$ and $U_{-\lambda}$, proving that $|U_\lambda| = |U_{-\lambda}|$.

**Lemma 5.6.** *If $h$ is an even natural number, $\lambda \in \mathbb{Z}_h$ and $\lambda^2 = 1$, then the eigenspaces $U_\lambda, U_{-\lambda} \leq \mathbb{Z}_h^{h^n}$ of $T_h^{\otimes n}$ have the same size: $|U_\lambda| = |U_{-\lambda}|$.*

*Proof.* Let consider the matrix $C_h^{(n)} = I_h \otimes \cdots \otimes I_h \otimes C_h = I_h^{\otimes(n-1)} \otimes C_h \in \mathbb{Z}_h^{h^n}$. The mixed product identity and Lemma 5.5 imply that $C_h^{(n)} T_h^{\otimes n} = -T_h^{\otimes n} C_h^{(n)}$:

$$\begin{aligned}T_h^{\otimes n} \cdot C_h^{(n)} &= (T_h \otimes \cdots \otimes T_h \otimes T_h) \cdot (I_h \otimes \cdots \otimes I_h \otimes C_h) \\ &= (T_h I_h) \otimes \cdots \otimes (T_h I_h) \otimes (T_h C_h) \\ &= (I_h T_h) \otimes \cdots \otimes (I_h T_h) \otimes (-C_h T_h) \\ &= -(I_h \otimes \cdots \otimes I_h \otimes C_h) \cdot (T_h \otimes \cdots \otimes T_h \otimes T_h) = -C_h^{(n)} \cdot T_h^{\otimes n}.\end{aligned}$$

We can use this fact to prove that if $\mathbf{v} \in U_\lambda$ then $C_h^{(n)} \mathbf{v} \in U_{-\lambda}$:

$$T_h^{\otimes n} C_h^{(n)} \mathbf{v} = -C_h^{(n)} T_h^{\otimes n} \mathbf{v} = -C_h^{(n)} \lambda \mathbf{v} = -\lambda C_h^{(n)} \mathbf{v}.$$

Therefore, we can define a map $\varphi \colon U_\lambda \to U_{-\lambda}$, $\mathbf{v} \mapsto C_h^{(n)} \mathbf{v}$.

Since $C_h$ is an upper triangular matrix with diagonal entries $\pm 1$, it has an inverse $C_h^{-1} \in \mathbb{Z}_h^{h \times h}$. Consequently, by the mixed product identity (1), the matrix $C_h^{(n)}$ also has an inverse (namely, $I_h \otimes \cdots \otimes I_h \otimes C_h^{-1}$). Taking the inverse of both sides of the equality $C_h^{(n)} T_h^{\otimes n} = -T_h^{\otimes n} C_h^{(n)}$ and recalling that $T_h^{\otimes n}$ is self-inverse,

we obtain $T_h^{\otimes n}\big(C_h^{(n)}\big)^{-1} = -\big(C_h^{(n)}\big)^{-1}T_h^{\otimes n}$. Then a similar argument to the one above leads us to infer that if $\mathbf{v} \in U_{-\lambda}$ then $\big(C_h^{(n)}\big)^{-1}\mathbf{v} \in U_\lambda$:

$$T_h^{\otimes n}\big(C_h^{(n)}\big)^{-1}\mathbf{v} = -\big(C_h^{(n)}\big)^{-1}T_h^{\otimes n}\mathbf{v} = -\big(C_h^{(n)}\big)^{-1}(-\lambda\mathbf{v}) = \lambda\big(C_h^{(n)}\big)^{-1}\mathbf{v}.$$

This allows us to define a map $\psi\colon U_{-\lambda} \to U_\lambda$, $\mathbf{v} \mapsto \big(C_h^{(n)}\big)^{-1}\mathbf{v}$. Clearly, $\varphi$ and $\psi$ are inverses of each other, so both are bijections, and this means that $|U_\lambda| = |U_{-\lambda}|$. □

Now we are ready to prove our main result about the eigenvectors of the Reed-Muller-Fourier transform. It is worth noting that if $h$ is even, then the number of eigenvectors does not depend on the eigenvalue $\lambda$ (as long as $\lambda^2 = 1$).

**Theorem 5.1.** *For every natural number $h$ and $\lambda \in \mathbb{Z}_h$ with $\lambda^2 = 1$, the eigenspace $U_\lambda\left(T_h^{\otimes n}\right) \leq \mathbb{Z}_h^{h^n}$ of the Reed-Muller-Fourier transform $T_h^{\otimes n}$ has cardinality*

$$\left|U_\lambda\left(T_h^{\otimes n}\right)\right| = \begin{cases} h^{\lfloor h^n/2\rfloor} \cdot \gcd(h, 1+\lambda), & \text{if } h \text{ is odd and } n \text{ is odd;} \\ h^{\lfloor h^n/2\rfloor} \cdot \gcd(h, 1-\lambda), & \text{if } h \text{ is odd and } n \text{ is even;} \\ h^{h^n/2}, & \text{if } h \text{ is even.} \end{cases}$$

*Proof.* Assume first that $h$ is even. Considering $U_\lambda$ and $U_{-\lambda}$ as additive subgroups of $\mathbb{Z}_h^{h^n}$, one of the isomorphism theorems (there seems to be no consensus on the numbering) yields $(U_\lambda + U_{-\lambda})/U_{-\lambda} \cong U_\lambda/(U_\lambda \cap U_{-\lambda})$, which implies with the help of lemmas 5.2 and 5.3 that

$$|U_\lambda| \cdot |U_{-\lambda}| = |U_\lambda + U_{-\lambda}| \cdot |U_\lambda \cap U_{-\lambda}| = \frac{h^{h^n}}{2^{h^n/2}} \cdot 2^{h^n/2} = h^{h^n}.$$

Then we may conclude from Lemma 5.6 that $|U_\lambda| = |U_{-\lambda}| = h^{h^n/2}$.

Now let us assume that $h$ is odd. Then we can apply Theorem 3.1, as $T_h^{\otimes n}$ is a triangular self-inverse matrix. Denoting the number of ones and zeros on the diagonal of $T_h^{\otimes n}$ by $m_1$ and $m_{-1}$, respectively, we see that

$$|U_\lambda| = \gcd(h, 1-\lambda)^{m_1} \cdot \gcd(h, -1-\lambda)^{m_{-1}} = \gcd(h, 1-\lambda)^{m_1} \cdot \gcd(h, 1+\lambda)^{m_{-1}}. \tag{10}$$

It is not hard to verify that the diagonal of $T_h^{\otimes n}$ is $(-1, 1, \ldots, 1, -1)$ if $n$ is odd and it is $(1, -1, \ldots, -1, 1)$ if $n$ is even (note that if $h$ is even then the diagonal entries of $T_h^{\otimes n}$ are still $\pm 1$, but not alternately; see tables 3 and 4). In the first case we have $m_1 = \lfloor h^n/2\rfloor, m_{-1} = \lceil h^n/2\rceil$, while in the second case we have $m_1 = \lceil h^n/2\rceil, m_{-1} = \lfloor h^n/2\rfloor$. Therefore, (10) gives with the help of Lemma 3.2 (note that $\lceil h^n/2\rceil = \lfloor h^n/2\rfloor + 1$),

$$|U_\lambda| = \gcd(h, 1-\lambda)^{\lfloor h^n/2\rfloor} \cdot \gcd(h, 1+\lambda)^{\lceil h^n/2\rceil} = h^{\lfloor h^n/2\rfloor} \cdot \gcd(h, 1+\lambda) \text{ if } 2 \nmid n,$$
$$|U_\lambda| = \gcd(h, 1-\lambda)^{\lceil h^n/2\rceil} \cdot \gcd(h, 1+\lambda)^{\lfloor h^n/2\rfloor} = h^{\lfloor h^n/2\rfloor} \cdot \gcd(h, 1-\lambda) \text{ if } 2 \mid n.$$

□

Now, we will conclude our study by proving Conjecture 1.1.

**Corollary 5.1.** *For all natural numbers $h \geq 2$ and $n \geq 1$, the number of fixed points of the Reed-Muller-Fourier transform on $n$-variable functions over $\mathbb{Z}_h$ is $h^{\lfloor h^n/2 \rfloor}$ if $n$ is odd, and it is $h^{\lceil h^n/2 \rceil}$ if $n$ is even.*

*Proof.* We apply Theorem 5.1 with $\lambda = 1$. If $h$ is even, then there is nothing to do; if $h$ is odd, then observe that $|U_\lambda| = h^{\lfloor h^n/2 \rfloor} \cdot \gcd(h, 1+1) = h^{\lfloor h^n/2 \rfloor} \cdot 1$ when $n$ is odd, and $|U_\lambda| = h^{\lfloor h^n/2 \rfloor} \cdot \gcd(h, 1-1) = h^{\lfloor h^n/2 \rfloor} \cdot h = h^{\lceil h^n/2 \rceil}$ when $n$ is even. □

# References


[1] Aburdene, M. F. and Goodman, T. J. The discrete Pascal transform and its applications. *IEEE Signal Processing Letters*, 12(7):493–495, 2005.

[2] Fine, N. J. Binomial coefficients modulo a prime. *The American Mathematical Monthly*, 54(10):589–592, 1947.

[3] Gibbs, J. E. Instant Fourier transform. *Electronics Letters*, 13(5):122–123, 1977.

[4] Goodman, T. J. and Aburdene, M. F. Interpolation using the discrete Pascal transform. In *40th Annual Conference on Information Sciences and Systems*, pages 1079–1083. 2006.

[5] Lucas, E. Theorie des Fonctions Numeriques Simplement Periodiques. *Amer. J. Math.*, 1(3):197–240, 1878.

[6] Moraga, C., Stanković, R. S., and Stanković, M. The Pascal triangle (1654), the Reed-Muller-Fourier transform (1992), and the discrete Pascal transform (2005). In *Proc. 46th IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, pages 229–234. 2016.

[7] Moraga, C., Stanković, R. S., Stanković, M., and Stojković, S. On fixed points of the Reed-Muller-Fourier transform. In *Proc. 47th IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, pages 55–60. 2017.

[8] Moraga, C., Stojković, S., and Stanković, R. On fixed points and cycles in the Reed Muller domain. In *Proc. 38th IEEE International Symposium on Multiple Valued Logic (ISMVL)*, pages 82–87. 2008.

[9] Muller, D. E. Application of Boolean algebra to switching circuit design and to error detection. *Transactions of the IRE Professional Group on Electronic Computers*, EC-3(3):6–12, 1954.

[10] Petkovšek, M., Wilf, H. S., and Zeilberger, D. $A = B$. A K Peters, Ltd., Wellesley, MA, 1996.

[11] Reed, I. A class of multiple-error-correcting codes and the decoding scheme. *Transactions of the IRE Professional Group on Information Theory*, 4(4):38–49, 1954.

[12] Sasao, T. and Butler, J. T. The eigenfunction of the Reed-Muller transformation. In *Proc. Workshop on Applications of the Reed Muller Expansion in Circuit Design and Representations and Methodology of Future Computing Technology*, pages 31–38. 2007.

[13] Stanković, R. S. Some remarks on Fourier transform and differential operators for digital functions. In *Proc. 22nd IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, pages 365–370. 1992.

[14] Stanković, R. S. The Reed-Muller-Fourier transform—computing methods and factorizations. In Seising, R. and Allende-Cid, H., editors, *Claudio Moraga: A Passion for Multi-Valued Logic and Soft Computing*, volume 349 of *Studies in Fuzziness and Soft Computing*, chapter 9, pages 121–151. Springer, 2017.

[15] Stanković, R. S., Astola, J. T., and Moraga, C. *Representation of Multiple-Valued Logic Functions*, volume 37 of *Synthesis Lectures on Digital Circuits and Systems*. Morgan & Claypool, 2012.

[16] Stanković, R. S., Moraga, C., and Astola, J. T. Reed-Muller expressions in the previous decade. *Journal of Multiple-Valued Logic and Soft Computing*, 10(1):5–28, 2004.

[17] Varsaki, E. E., Fotopoulos, V. E., and Skodras, A. N. On the use of the discrete Pascal transform in hiding data in images. In *Proceedings of the SPIE — Optics, Photonics, and Digital Technologies for Multimedia Applications*, volume 7723, page 77230L. 2010.

[18] Waldhauser, T. On the number of fixed points of the Reed-Muller-Fourier transform. Submitted to Proc. 48th IEEE International Symposium on Multiple-Valued Logic (ISMVL).

[19] Zhegalkin, I. I. On the techniques of calculating sentences in symbolic logic. *Math. Sb.*, 34:9–28, 1927. Russian.

[20] Zhegalkin, I. I. Arithmetic representations for symbolic logic. *Math. Sb.*, 35:311–377, 1928. Russian.

*Received 4th May 2018*

Table 1: The matrix $P_8$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -3 & 3 & -1 & 0 & 0 & 0 & 0 \\ 1 & -4 & 6 & -4 & 1 & 0 & 0 & 0 \\ 1 & -5 & 10 & -10 & 5 & -1 & 0 & 0 \\ 1 & -6 & 15 & -20 & 15 & -6 & 1 & 0 \\ 1 & -7 & 21 & -35 & 35 & -21 & 7 & -1 \end{pmatrix}$$

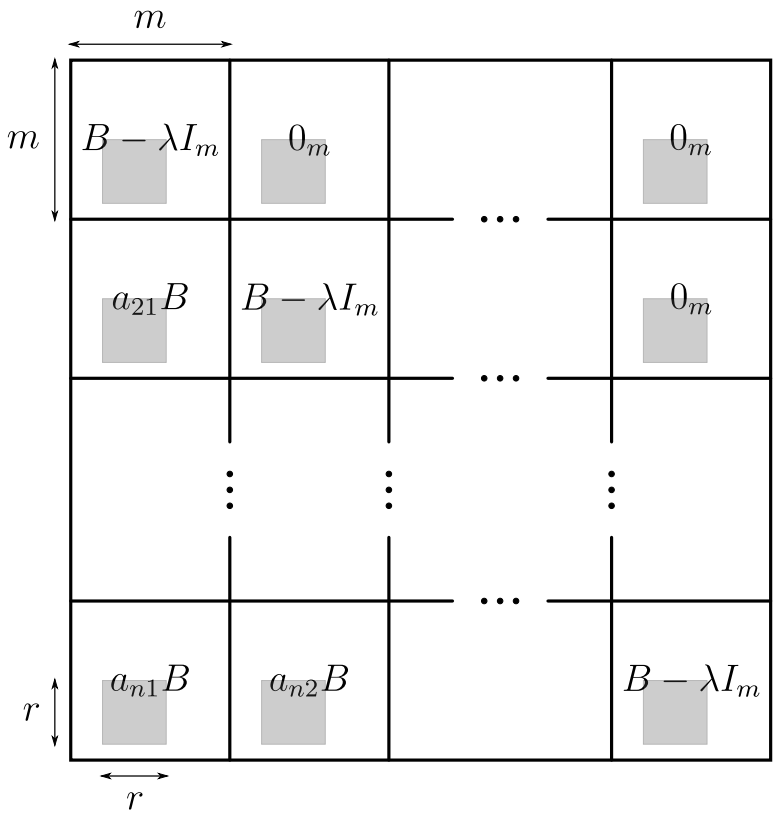



Figure 1: The matrix $A \otimes B - \lambda I_{nm}$ in the proof of Lemma 2.1

Table 2: The matrix $T_8$

$$\begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -3 & 1 & 0 & 0 & 0 & 0 \\ -1 & 4 & -6 & 4 & -1 & 0 & 0 & 0 \\ -1 & 5 & -10 & 10 & -5 & 1 & 0 & 0 \\ -1 & 6 & -15 & 20 & -15 & 6 & -1 & 0 \\ -1 & 7 & -21 & 35 & -35 & 21 & -7 & 1 \end{pmatrix}$$

Table 3: The matrix $T_2^{\otimes 2}$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

Table 4: The matrix $T_2^{\otimes 3}$

$$\begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 1 & 0 & -1 & 0 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{pmatrix}$$

Table 5: The matrix $T_3^{\otimes 2}$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & -1 & 2 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & 0 & -2 & 2 & 0 & 1 & -1 & 0 \\ 1 & -2 & 1 & -2 & 4 & -2 & 1 & -2 & 1 \end{pmatrix}$$

Table 6: The matrix $A_8$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 3 & -3 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 3 & -3 & 1 \end{pmatrix}$$

Table 7: The matrix $C_8$

$$\begin{pmatrix} -1 & 14 & -84 & 280 & -560 & 672 & -448 & 128 \\ 0 & 1 & -12 & 60 & -160 & 240 & -192 & 64 \\ 0 & 0 & -1 & 10 & -40 & 80 & -80 & 32 \\ 0 & 0 & 0 & 1 & -8 & 24 & -32 & 16 \\ 0 & 0 & 0 & 0 & -1 & 6 & -12 & 8 \\ 0 & 0 & 0 & 0 & 0 & 1 & -4 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Table 8: Sizes of eigenspaces of $T_h$ for $h \leq 12$

| | | $h$ | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| $\lambda$ | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 1 | 2 | 3 | $2^4$ | $5^2$ | $2^3 3^3$ | $7^3$ | $2^{12}$ | $3^8$ | $2^5 5^5$ | $11^5$ | $2^{12} 3^6$ |
| | 2 | | $3^2$ | 1 | 1 | $3^3$ | 1 | 1 | $3^5$ | 1 | 1 | $3^6$ |
| | 3 | | | $2^4$ | 1 | $2^3$ | 1 | $2^{12}$ | 1 | $2^5$ | 1 | $2^{12}$ |
| | 4 | | | | $5^3$ | $3^3$ | 1 | 1 | $3^4$ | $5^5$ | 1 | $3^6$ |
| | 5 | | | | | $2^3 3^3$ | 1 | $2^{12}$ | $3^5$ | $2^5$ | 1 | $2^{12} 3^6$ |
| | 6 | | | | | | $7^4$ | 1 | 1 | $5^5$ | 1 | 1 |
| | 7 | | | | | | | $2^{12}$ | $3^4$ | $2^5$ | 1 | $2^{12} 3^6$ |
| | 8 | | | | | | | | $3^{10}$ | 1 | 1 | $3^6$ |
| | 9 | | | | | | | | | $2^5 5^5$ | 1 | $2^{12}$ |
| | 10 | | | | | | | | | | $11^6$ | $3^6$ |
| | 11 | | | | | | | | | | | $2^{12} 3^6$ |

CONTENTS

In Memoriam Csanád Imreh . . . . . 757
*Ron Adar and Leah Epstein*: The Metric Dimension of Two-Dimensional Extended Meshes . . . . . 761
*József Békési*: Regional Multicriteria and Multimodal Route Planning System for Public Transportation: A Case Study . . . . . 773
*József Békési, Balázs Dávid, and Miklós Krész*: Integrated Vehicle Scheduling and Vehicle Assignment . . . . . 783
*Gábor Berend*: $\ell_1$ Regularization of Word Embeddings for Multi-Word Expression Identification . . . . . 801
*Csilla Bujtás and Zsolt Tuza*: Partition-Crossing Hypergraphs . . . . . 815
*József Dombi and Tamás Jónás*: Approximations to the Normal Probability Distribution Function using Operators of Continuous-valued Logic . . . . 829
*György Dósa and Leah Epstein*: The Convergence Time for Selfish Bin Packing 853
*Zoltán Fülöp and Zsolt Gazdag*: Weighted Languages Recognizable by Weighted Tree Automata . . . . . 867
*Kitti Gelle and Szabolcs Iván*: DFS is Unsparsable and Lookahead Can Help in Maximal Matching . . . . . 887
*Tamás Gergely, Gergő Balogh, Ferenc Horváth, Béla Vancsics, Árpád Beszédes, Tibor Gyimóthy*: Analysis of Static and Dynamic Test-to-code Traceability Information . . . . . 903
*András London and András Pluhár*: Spanning Tree Game as Prim Would Have Played . . . . . 921
*Judit Nagy-György*: On the Advice Complexity of Coloring Bipartite Graphs and Two-Colorable Hypergraphs . . . . . 929
*Kálmán Palágyi*: How Sufficient Conditions are Related for Topology-Preserving Reductions . . . . . 939
*Tamás Waldhauser*: On Eigenvectors of the Pascal and Reed-Muller-Fourier Transforms . . . . . 959

**ISSN 0324—721 X**

Felelős szerkesztő és kiadó: Csendes Tibor